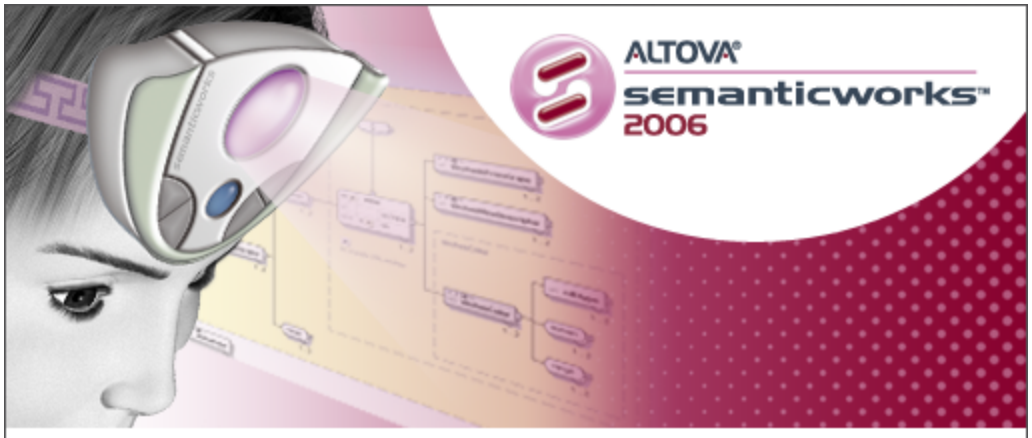
User and Reference Manual



Copyright ©1998-2005 Altova GmbH. All rights reserved. Use of this software is governed by and subject to an Altova software license agreement. XMLSpy, MapForce, StyleVision, SemanticWorks, SchemaAgent, UModel, DiffDog, Authentic, AltovaXML, and ALTOVA are trademarks and/or registered trademarks of Altova GmbH. Patent(s) pending.

XML, XSL, XHTML, and W3C are trademarks (registered in numerous countries) of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions, MIT, INRIA, and Keio. UNICODE and the Unicode Logo are trademarks of Unicode, Inc. This software contains material that is ©1994-1998 Dundas Software Ltd., all rights reserved. The Sentry Spelling Checker Engine - ©2000 Wintertree Software Inc. , STLport ©1999,2000 Boris Fomitchev, © 1994 Hewlett-Packard Company, ©1996,97 Silicon Graphics Computer Systems, Inc., ©1997 Moscow Center for SPARC Technology. Scintilla Copyright ©1998-2002 by Neil Hodgson <neilh@scintilla.org>. ANTLR Copyright ©1989-2005 by Terence Parr <www.antlr.org>. All other names or trademarks are the property of their respective owners.

ALTOVA®

Altova SemanticWorks User Manual

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Published: 2006

© 2006 Altova GmbH

Table of Contents

1 SemanticWorks 2006 3

2 About this Documentation 6

3 Introduction 8

3.1 Product Features 9
3.2 Interface 11
3.2.1 Main Window 13
3.2.2 Overview Window 16
3.2.3 Errors Window 17
3.3 Overview of Usage 19

4 Tutorial 24

4.1 OWL Lite Ontology 26
4.1.1 Creating a New Ontology 27
4.1.2 Declaring Namespaces 30
4.1.3 Creating Classes 32
4.1.4 Creating the Class Hierarchy 34
4.1.5 Defining Properties 38
4.1.6 Declaring Instances 44
4.1.7 Declaring AllDifferent Instances 49
4.2 OWL DL Ontology 51
4.2.1 Setting Up an OWL DL Ontology 52
4.2.2 Creating the Classes 54
4.2.3 Instances as Class Enumerations 56
4.2.4 Defining the Properties 58
4.2.5 Describing Classes and Their Instances 60
4.3 RDF Documents 65
4.3.1 Instances for an OWL DL Ontology 66
Creating a New RDF Document 66
Referencing the Ontology 66
Making the RDF statements 68

4.3.2 Creating a Dublin Core (DC) Document ... 71
Referencing the DC Ontology ... 71
Creating the DC Metadata ... 74

5 User Reference 78

5.1 Toolbar Icons ... 79
5.2 Icons in Detail View ... 84
5.3 File Menu ... 87
5.4 Edit Menu ... 89
5.5 View Menu ... 90
5.6 RDF/OWL Menu ... 92
5.7 Tools Menu ... 94
5.7.1 Options ... 95
5.7.2 Namespace Imports for RDF ... 97
5.7.3 URIref Prefixes and Namespaces ... 99
5.7.4 Base URI ... 100
5.8 Window Menu ... 101
5.9 Help Menu ... 102
5.10 Usage Issues ... 103

6 Conformance 106

7 License Information 108

7.1 Electronic Software Distribution ... 109
7.2 License Metering ... 110
7.3 Copyright ... 111
7.4 Altova End User License Agreement ... 112

Index

Chapter 1

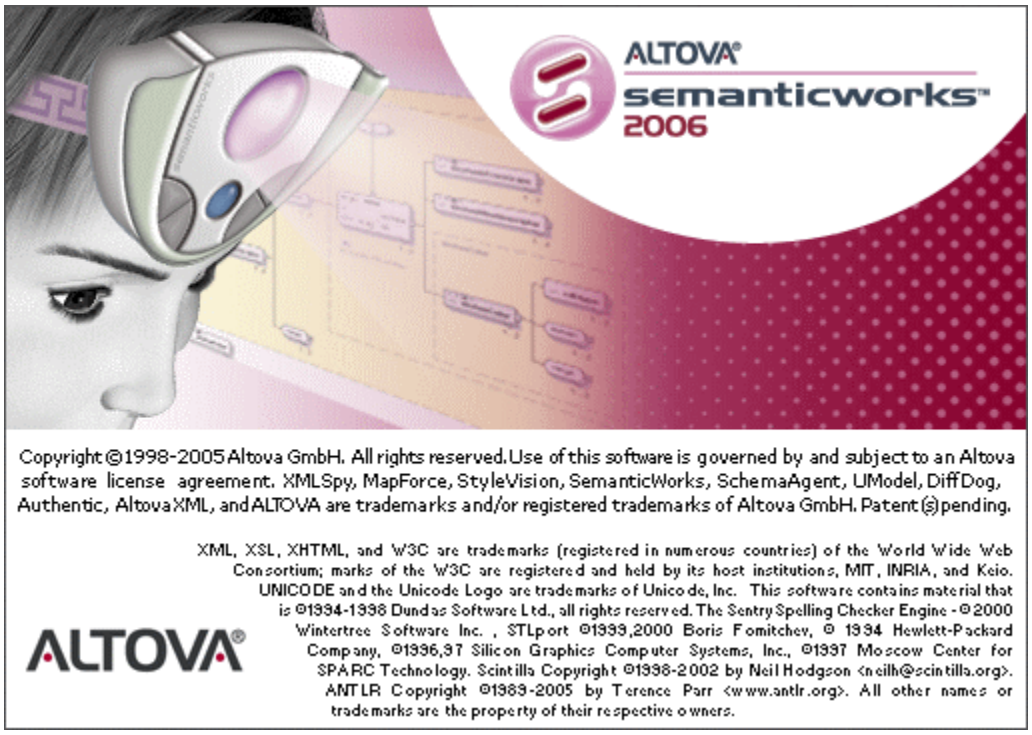
SemanticWorks 2006

1 SemanticWorks 2006

Altova® SemanticWorks™ 2006 is an RDF document editor and ontology development IDE. It enables you to:

- Graphically create and edit RDF documents, RDF Schema documents, and OWL ontologies.
- Check the syntax and semantics of ontologies as you edit them, and the syntax of RDF documents.
- Convert graphically created ontologies into the RDF/XML and N-Triples formats.

With **Altova® SemanticWorks™ 2006**, therefore, besides being able to edit RDF documents in a GUI and check its syntax, you can design RDF Schema and OWL ontologies using a graphical design view, check the syntax of any RDF Schema or OWL ontology and the semantics of OWL Lite and OWL DL ontologies, and export ontologies in the RDF/XML and N-Triples formats.


ALTOVA®
semanticworks™
2006
Copyright ©1998-2005 Altova GmbH. All rights reserved. Use of this software is governed by and subject to an Altova software license agreement. XMLSpy, MapForce, StyleVision, SemanticWorks, SchemaAgent, UModel, DiffDog, Authentic, AltovaXML, and ALTOVA are trademarks and/or registered trademarks of Altova GmbH. Patent(s) pending.
XML, XSL, XHTML, and W3C are trademarks (registered in numerous countries) of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions, MIT, INRIA, and Keio. UNICODE and the Unicode Logo are trademarks of Unicode, Inc. This software contains material that is ©1994-1998 Dundas Software Ltd., all rights reserved. The Sentry Spelling Checker Engine - ©2000 Wintertree Software Inc. , STLport ©1999,2000 Boris Fomitchev, © 1994 Hewlett-Packard Company, ©1996,97 Silicon Graphics Computer Systems, Inc., ©1997 Moscow Center for SPARC Technology. Scintilla Copyright ©1998-2002 by Neil Hodgson <neilh@scintilla.org>. ANTLR Copyright ©1989-2005 by Terence Parr <www.antlr.org>. All other names or trademarks are the property of their respective owners.
ALTOVA®

Chapter 2

About this Documentation

2 About this Documentation

This documentation is the user manual delivered with SemanticWorks. It is available as the built-in Help system of SemanticWorks, can be viewed online at the Altova website, and can also be downloaded as a PDF, which you can print.

The user manual is organized into the following sections:

- Introduction
- Tutorial
- User Reference
- Conformance

We suggest you read the Introduction first in order to get an overview of SemanticWorks features and general usage. You should then go through the tutorial to get hands-on experience of creating OWL Lite and OWL DL ontologies, and of creating and editing RDF documents. For subsequent reference, use the user reference section, which provides a description of all toolbar icons and menu commands.

Should you have any question or problem related to SemanticWorks, the following support options are available:

1. Check the Help file (this documentation). The Help file contains a full text-search feature, besides being fully indexed.
2. Check the FAQs and Discussion Forum at the Altova Website.
3. Contact Altova's Support Center.

© 2006 Altova GmbH

Chapter 3

Introduction

3 Introduction

This Introduction is organized into the following sections:

- Product Features: Lists the main product features of SemanticWorks 2006. Read through this section to get an overview of the capabilities of SemanticWorks.
- Interface: Describes the SemanticWorks GUI. This description helps familiarize you with the various views and windows of SemanticWorks, and shows you how they are used.
- Overview of Usage: Provides a methodological approach to using SemanticWorks. Lists the steps you would typically take when creating or editing an ontology and an RDF document in SemanticWorks.
- Terminology: Lists key terms used in SemanticWorks and this documentation together with their meanings. Reading through this section will also provide you with a quick summary of key RDF and OWL concepts.

© 2006 Altova GmbH

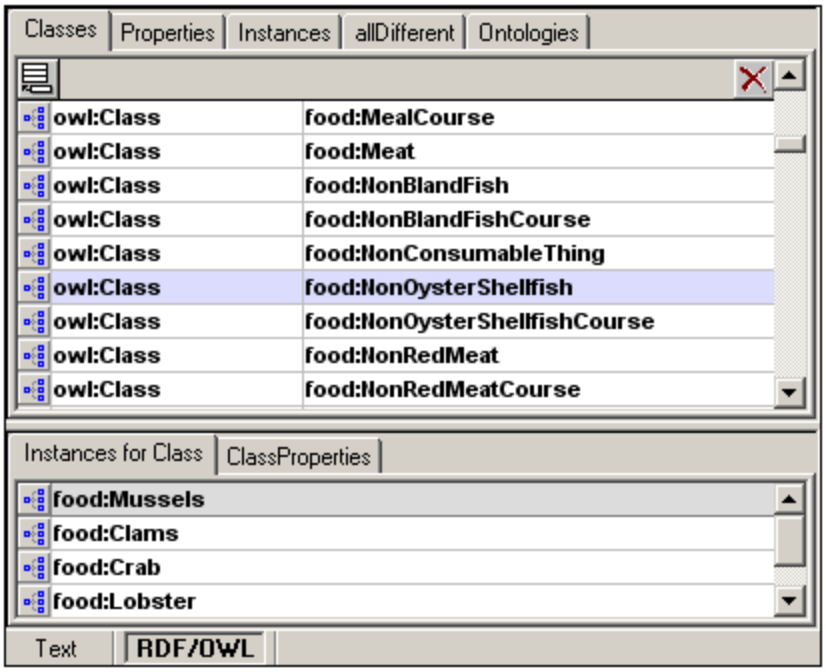
3.1 Product Features

The main features of SemanticWorks 2006 are listed below.

Editing RDF documents
In SemanticWorks, RDF documents can be created and edited graphically in SemanticWorks's RDF/OWL View. An RDF resource is defined by graphically associating it with a predicate, and then associating the predicate with a resource object or literal value. Resources are made available for selection in the GUI by referencing an ontology. A mechanism for declaring namespaces and prefixes enables URIrefs for RDF resources to be assigned flexibly and accurately. SemanticWorks also checks the syntax of RDF documents. Alternatively to editing RDF documents in the graphical RDF/OWL View, RDF documents can be edited directly in Text View, using either RDF/XML notation or N-Triples notation.

Editing ontologies
SemanticWorks offers ontology editing capability in a graphical user interface and in a text interface.

- The graphical RDF/OWL View enables you to easily create and edit RDF Schema and OWL ontologies by allowing you to insert items into a graphical representation of the ontology.
- The ontology level can be changed at any time while editing, enabling you to change levels according to editing needs.
- Syntax checks can be carried out for RDF Schema, OWL Lite, OWL DL, and OWL Full ontologies. Semantics checks can be carried out on OWL Lite and OWL DL documents. These checks can be made while you edit, thus enabling you to easily maintain the validity of the ontology as you build it.
- The ontology document's classes, properties, instances (aka individuals), all-different items, and ontologies can be viewed in separate tabs (*screenshot below*). These tabs provide an overview of each of these categories. A subsidiary pane displays related information (*screenshot below*). For example, selecting a class in the Classes Overview causes the instances and properties of that class to be displayed in the subsidiary pane.



- Clicking the Detail View button (*see Main Window*) of an ontology item in the Overview, changes the view to a detailed view of that item's relationships (*see screenshot*).
- In Detail View, relationships are indicated by a range of intuitive icons which are inserted into the ontology via a context menu.
- Relationships in Detail View can be expanded and collapsed at multiple levels to provide easily customizable views of ontologies or specific parts of ontologies.
- Prefixes of URIrefs used in an ontology can be conveniently set in a special table accessed via the GUI.
- The display of blank nodes (anonymous classes) can be toggled on and off.
- Text View enables direct text editing of the ontology document.

Checking documents

An RDF, RDF Schema, OWL Lite, OWL DL, or OWL Full document can be checked for syntax according to the rules of the relevant specification/s. Additionally, OWL Lite and OWL DL documents can be checked for correct semantics (according to the rules of OWL Lite and OWL DL, respectively). Errors are listed in the Errors Window, and each error has one or more links to the incorrect item/s in the current view (Text View or RDF/OWL View) of that document.

Other major features

SemanticWorks offers the following additional features.

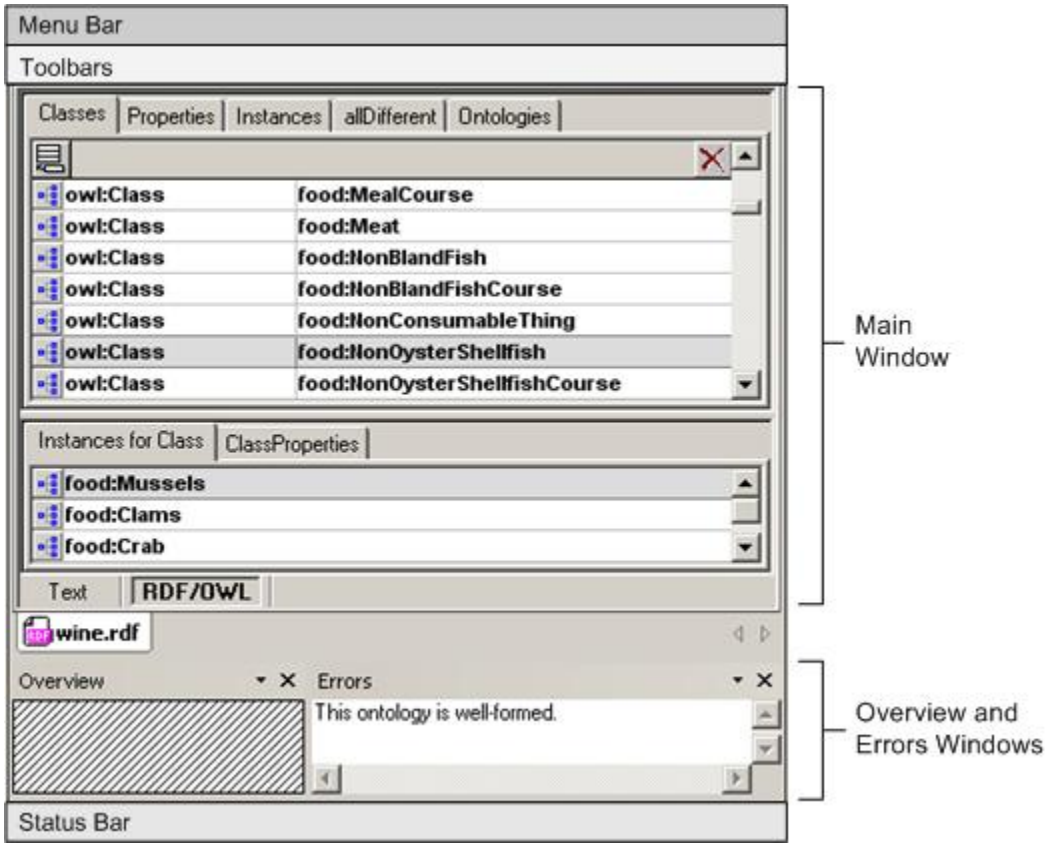
- Imports can be reloaded at the click of a button whenever required.
- Ontologies can be saved as `.rdf`, `.rdfs`, or `.owl` files and can be exported in their RDF/XML and N-Triples formats.
- Multiple ontologies can be edited concurrently in multiple windows.
- A large range of customization options enables the application interface to be flexibly customized. Options range from GUI font selection to choosing document encoding.
- The graphical Detail View of ontology items can be printed as well as saved as an image.

 © 2006 Altova GmbH

3.2 Interface

The SemanticWorks application interface consists of: (i) a top part consisting of a Menu Bar and Toolbars; (ii) the windows area; and (iii) a bottom part consisting of the Status Bar. *See screenshot below.* The windows areas consist of three windows:

- The Main Window,
- The Overview Window, and
- The Errors Window.

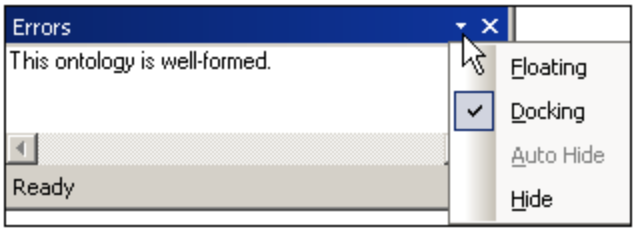


The **Menu Bar** contains the various menus, and each menu, with its menu items, are described in separate sections in the User Reference. The **toolbars** are located below the Menu Bar, and all toolbar icons are described in the Toolbar Icons section in the User Reference.

The Main Window, Overview Window, and Errors Window are described in more detail in the subsections of this section. The Overview Window and Errors Window can be docked within the application window or can float freely. For details about how to change the position of the Overview Window and Errors Window, see under the next heading.

Note: Multiple documents can be open at a time, and any one can be made the active document by clicking its tab label at the bottom of the Main Window. Text can be copied between the Text Views of documents, but objects in RDF/OWL Views cannot be copied.

Moving, positioning, and hiding the Overview Window and Errors Window
The Overview Window and Errors Window can each be docked within the application window or they can each float freely as independent windows.



To make the Overview Window or Errors Window float, do one of the following:

- Drag the window's title bar out of its docked position, so that the window floats, or
- Click the down-pointing arrowhead at the right-hand side of the window's title bar and select Floating.

To reposition the Overview Window or Errors Window relative to the Main Window or the other window (that is, to dock it), do the following:

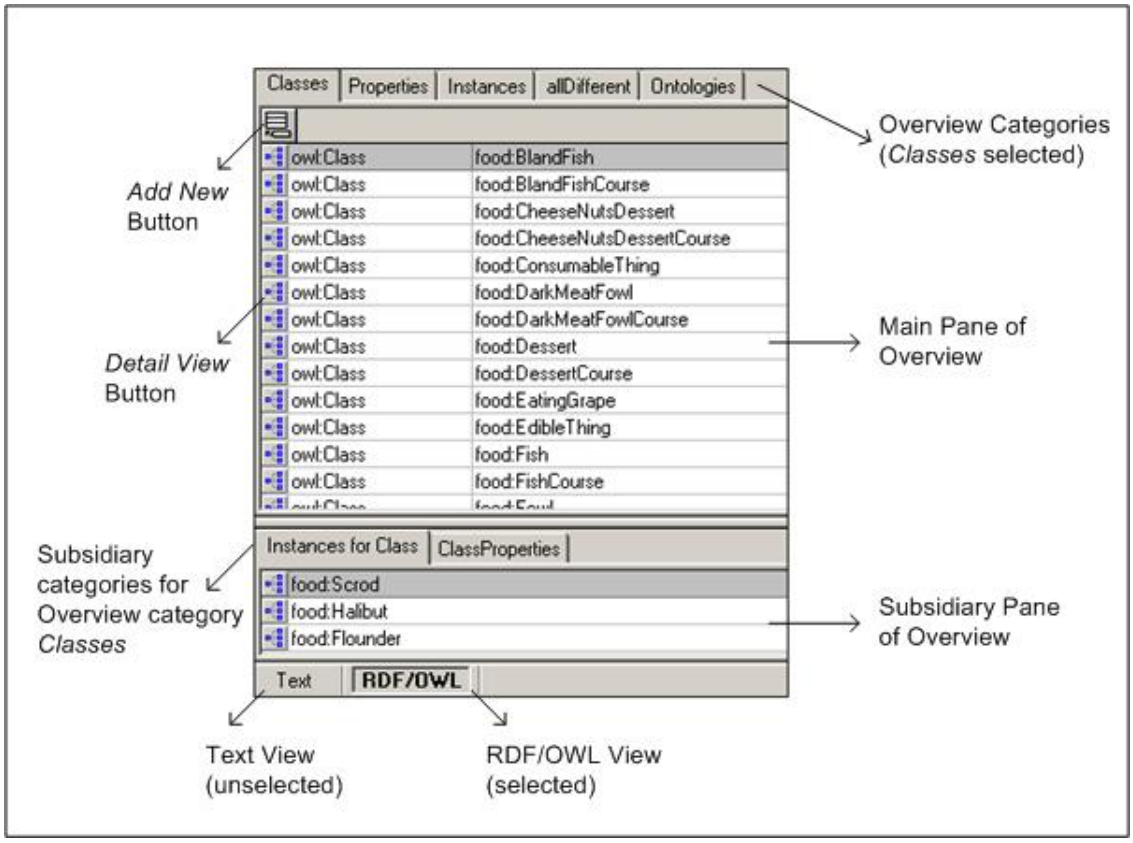
- Drag the window's title bar into the application window till two sets of four blue arrows appear (an inner set and an outer set). Drop the window on one of the four inner arrows or one of the four outer arrows. If you drop it on an inner arrow, the window will dock within the application window and relative to the window over which it was dragged. If you drop it on an outer arrow, the window will be placed along one of the four inside edges of the application window.

When the Overview Window or Errors Window is docked, clicking on the down-pointing arrowhead (*see screenshot above*) also provides the option of hiding the window (menu option **Hide**). Clicking the Close button (at right-hand side of the Overview Window or Errors Window title bar), closes the window. To reopen the Overview Window or Errors Window, select **View | Toolbars | Overview** or **View | Toolbars | Errors Window**, respectively.

 © 2006 Altova GmbH

3.2.1 Main Window

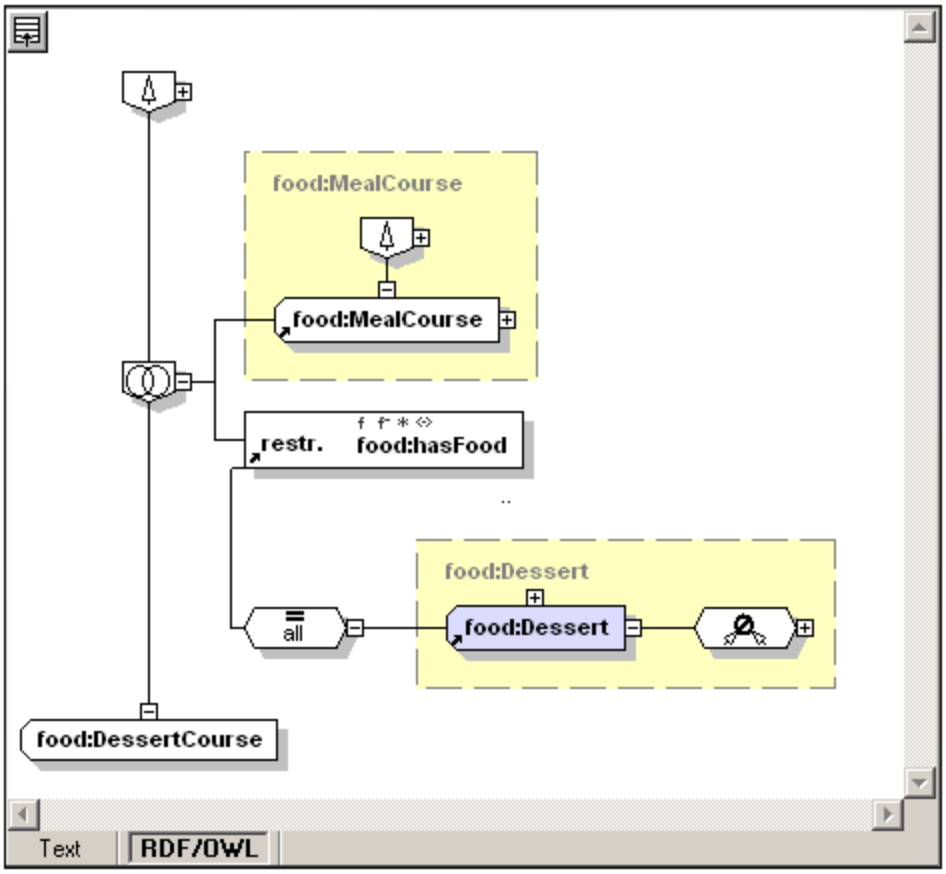
SemanticWorks documents are opened in the Main Window, and are viewed and edited in the Main Window. The Main Window has two views, Text View and RDF/OWL View *(screenshots below)*, between which you can switch by clicking the respective tabs.



RDF/OWL View window, showing Overview (main pane, with subsidiary pane below). To switch from Overview to Detail View of an item, click the Detail View button to the left of that item.

RDF/OWL View

RDF/OWL View provides (i) an Overview of the document, and (ii) a Detail View of an item listed in the Overview. To switch from Overview to the Detail View of an item, click the Detail View icon to the left of that item's Overview listing. Notice that the entire Overview (main pane and subsidiary pane) is replaced by Detail View (*screenshot below*). To switch from the Detail View of an item back to Overview, click the Overview icon located at the top left of Detail View .



The **Overview** of ontologies is structured into five categories of items (*see first screenshot in this section*):

- ***Classes***, which lists all ontology classes. If the Show Blank Nodes option (**View | Show Blank Nodes**) is selected, then anonymous classes are also listed. When a class is selected in the main pane, then the subsidiary pane shows (i) the properties of the class, and (ii) the instances of the class.
- ***Properties***, which lists all properties in the ontology. When a property is selected in the main pane, then the domain of that property is displayed in the subsidiary pane.
- ***Instances*** (aka individuals), which lists all the ontology's instances of classes.
- ***All-Different*** items, which lists the `owl:AllDifferent` items in the ontology.
- ***Ontologies***, which lists all ontologies in the document, including imported and prior-version ontologies.

Each category has a tab in the main pane of the Overview, with tab labels located at the top of the main pane. To view the items in a particular category, click that category's tab label. When either the Classes or Properties tab is selected, and an individual class or property, respectively, in that tab is selected, additional information related to the selected class or property is displayed in a subsidiary pane located below the main pane. When a class in the Classes tab is selected, its individuals and properties can be viewed in the subsidiary pane. When a property in the Properties tab is selected, its domain is displayed in the subsidiary pane.

The Overview of RDF documents lists items in a single category: ***Resources***.

Note: If an ontology imports other ontologies, then the classes, properties, instances, and all-different items of the imported ontologies are also displayed. If an RDF document correctly

 © 2006 Altova GmbH

references an ontology, the items of the ontology are displayed as resources in the GUI.

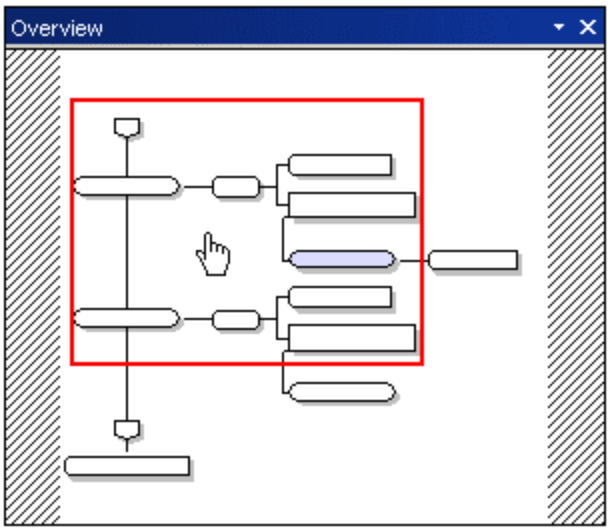
In **Detail View**, you add or edit the details of an ontology item. Items are added by right-clicking an item and selecting the new item to insert from a context menu. Items that can be inserted are listed and described in the section, Icons in Detail View.

Text View
In Text View, you can display and edit a document in its RDF/XML notation (*screenshot below*) as well as in N-Triples notation.

Text View supports syntax coloring, line numbering, source folding of elements, and standard GUI editing features, such as cut-and-paste and drag-and-drop.

3.2.2 Overview Window

The Overview Window is populated when the Main Window is in Detail View. It shows a miniature of the graphic currently in Detail View and the Detail View viewport (that is, the area of the Detail View graph that is currently visible in the Detail View window). *See screenshot below*.



The area of the Detail View graph currently displayed in the viewport is the area within the red rectangle. You can move other parts of the graph into the viewport by placing the cursor inside the red rectangle (the cursor becomes a pointing hand) and dragging the rectangle over to the part of the graph you want displayed in the viewport; that part of the graph will now be displayed in the Detail View window (the viewport).

© 2006 Altova GmbH

3.2.3 Errors Window

The Errors Window displays the results of the syntax and semantic checks.

- The **syntax check** is executed when the menu command **RDF/OWL | Syntax Check** is selected or when the toolbar icon is clicked.
- The **semantics check** (for OWL Lite and OWL DL ontologies) is executed when the menu command **RDF/OWL | Semantics Check** is selected or when the toolbar icon is clicked. The semantics check always includes a syntax check.

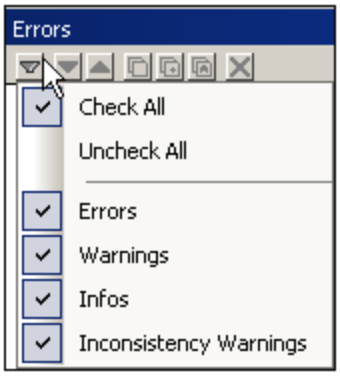
If the test returns a positive result, a message about the ontology being well-formed (syntax check) or partially consistent (semantics check) is displayed in the Errors Window. The semantics check (for OWL Lite and OWL DL ontologies) is a partial consistency check and returns a positive result (partially consistent) if no error or inconsistency is found.

Below the title bar of the Errors Window is a bar containing buttons that allow you to: (i) configure the display of messages in the Errors Window; (ii) navigate the messages in the Errors Window; and save messages to the clipboard. See screenshot below.

These buttons are described below, and a tooltip for each is displayed when the cursor is placed over it.

Filters and display

Clicking the **Filter** button (leftmost button), pops up a menu (screenshot below), in which you can select what messages are displayed in the Errors Window.

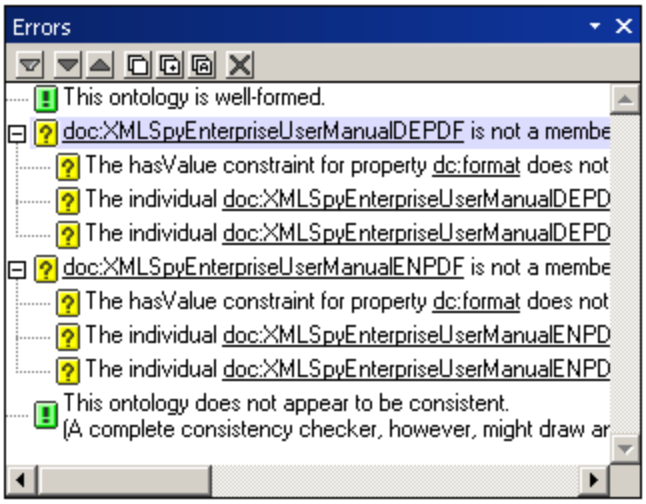


In this menu, you can select whether errors, warnings, information, and/or inconsistency warnings should be displayed in the Errors Window. To select a particular message type for display, select it so that it is checked. The Check All option causes all message types to be selected for display; the Uncheck All option deselects all message types.

The **Clear** button, clears all messages currently displayed.

Navigation

The **Next** and **Previous** buttons (second and third from left) enables you to navigate up and down the list of messages, respectively, one message at a time. When a message is selected, this is indicated by its being highlighted (*see screenshot below*).


Errors
This ontology is well-formed.
doc:XMLSpyEnterpriseUserManualDEPDF is not a membe
The hasValue constraint for property dc:format does not
The individual doc:XMLSpyEnterpriseUserManualDEPD
The individual doc:XMLSpyEnterpriseUserManualDEPD
doc:XMLSpyEnterpriseUserManualENPDF is not a membe
The hasValue constraint for property dc:format does not
The individual doc:XMLSpyEnterpriseUserManualENPD
The individual doc:XMLSpyEnterpriseUserManualENPD
This ontology does not appear to be consistent.
(A complete consistency checker, however, might draw ar

You can also select a message by clicking it. Selecting a message is useful if you wish to save a particular message to the clipboard.

Copying to clipboard
There are three ways in which messages can be copied to the clipboard: (i) copy the selected line; (ii) copy the selected line and its children; and (iii) copy all messages. The corresponding buttons are the fourth, fifth, and sixth from left. Placing the cursor over these buttons causes a tooltip for that button to be displayed.

 © 2006 Altova GmbH

3.3 Overview of Usage

This section broadly describes how SemanticWorks is to be used to:

- Create and edit ontologies
- Create and edit RDF documents

Creating and editing ontologies
When creating or editing ontologies with SemanticWorks, the broad usage procedure is as follows:

1. Create a new ontology document or load an existing ontology document into SemanticWorks.
2. Edit the document in RDF/OWL View.
3. Within SemanticWorks, check document syntax and/or semantics against RDF Schema, OWL Lite, OWL DL, or OWL Full specifications.
4. Save the document as a `.rdf`, `.rdfs`, or `.owl` file.
5. If required, export the document as an N-Triples (`.nt`) or XML (`.xml`) file.

Steps 1, 4, and 5 in the above process are straightforward. In this section, we briefly discuss how ontologies can be edited in RDF/OWL View (Step 2 above) and checked for correct syntax and semantics (Step 3).

Editing ontologies in RDF/OWL View
Ontologies are best edited in RDF/OWL View. Text View should be used to check the serialization, in XML format, of the ontology graph that was created or edited in RDF/OWL View. Additionally, Text View can be used to make minor modifications to the XML serialization so that this suits user preferences. However, most editing should be done in RDF/OWL View since this provides a graphical, intuitive, and fast way to edit ontologies.

Editing an ontology in RDF/OWL View involves the following processes. There is no strict sequence to be followed, and you will likely find yourself revisiting previous steps and revising various ontology items.

- *Declaring namespaces and their prefixes*. This is done at the document level (in the URIref Prefixes dialog (**Tools | URIref Prefixes**)). Declaring namespaces is important because they are used to identify ontology constructs, items from various vocabularies, and user-defined resources. The RDF, RDFS, and OWL namespaces are declared by default when a new ontology is created.
- *Select the ontology level*. You do this using the menu command **RDF/OWL | RDF/OWL Level**. Selecting the required level is important because: (i) the choice of constructs made available in the GUI, and (ii) the syntax and semantics checks done by SemanticWorks are based on this selection.
- *Setting up the ontology header*. The ontology header is created at the ontology level and is optional. It is useful when you wish to import one or more ontologies into the current ontology, or when you wish to record a prior version of the ontology. You create a new ontology in the Ontologies tab of Overview, then switch to Detail View and define the ontology using Detail View editing mechanisms.
- *Creating new ontology items*. Ontology items are classes, properties, instances, AllDifferent items, and ontologies. Each such item must first be created in the appropriate Overview tab. Only after the relevant items have been created, should you go to the Detail View of an item to either define attributes of the item (e.g. create restrictions for properties), or define relationships with other items (e.g. define an intersection of classes).
- *Defining and editing items in Detail View*. Definitions for ontology items are created and

edited in Detail View by selecting the required properties or relationships from a context menu. SemanticWorks will make only those constructs available that are allowed according to the ontology level you have selected.

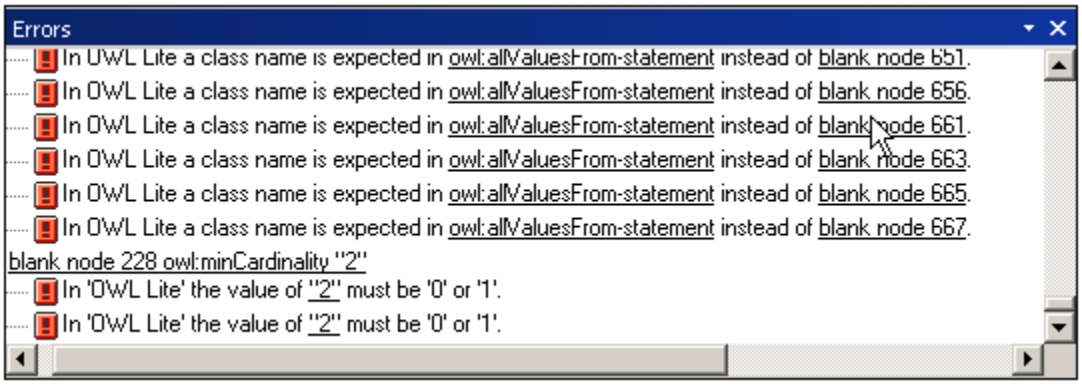
Checking the syntax and semantics of ontologies

When you edit an ontology in RDF/OWL View, what is created is a graph of the ontology. This graph is serialized in RDF/XML format—which is what is displayed in Text View. The syntax of this document serialization can be checked for conformance. Additionally, OWL Lite and OWL DL documents can be checked for correct semantics against the OWL Lite and OWL DL specifications. Note, however, that the semantics check in SemanticWorks is a partial consistency check. The significance of this is explained in the descriptions of the commands Show Possible Inconsistencies and Semantics Check.

To check the syntax of ontology documents and the semantics of OWL Lite and OWL DL documents, you do the following:

- Select the required specification against which the ontology is to be checked (**RDF/OWL | RDF/OWL Level**).
- Click the Syntax Check or Semantics Check command (**RDF/OWL** menu) or button (in the Toolbar).

If errors are detected, these are reported in the Errors Window, with each error including a link (or links) to the relevant ontology item (*screenshot below*).


Errors
In OWL Lite a class name is expected in owl:allValuesFrom-statement instead of blank node 651.
In OWL Lite a class name is expected in owl:allValuesFrom-statement instead of blank node 656.
In OWL Lite a class name is expected in owl:allValuesFrom-statement instead of blank node 661.
In OWL Lite a class name is expected in owl:allValuesFrom-statement instead of blank node 663.
In OWL Lite a class name is expected in owl:allValuesFrom-statement instead of blank node 665.
In OWL Lite a class name is expected in owl:allValuesFrom-statement instead of blank node 667.
blank node 228 owl:minCardinality "2"
In 'OWL Lite' the value of "2" must be '0' or '1'.
In 'OWL Lite' the value of "2" must be '0' or '1'.

Creating and editing RDF documents

In RDF/OWL View, when the RDF/OWL level has been set to RDF, resources are listed in the Overview pane of RDF/OWL View. The resources listed here are of two types: (i) those that are made available from an ontology, and (ii) those that you create (or that are present in the RDF document you are editing).

In order to make resources from an ontology available in the Resources Overview, you do the following:

1. Import the namespaces used in the ontology.
2. Declare the namespaces required in the RDF document.

In the Resources Overview, you create resources as required, and name them. Next, in the Detail View of each resource, you insert predicates, either by entering them or by selecting them from a dropdown list of available resources (which include resources made available from an ontology). The objects of RDF statements can also be inserted either by entering the name of an RDF resource or selecting one from a list of available resources, or by entering a literal value for the object. You can check the syntax of the RDF document at any time.

 © 2006 Altova GmbH

See the RDF Document tutorial for a detailed description of the steps listed above.

Alternatively, you can create and edit RDF documents directly in Text View, using RDF/XML or N-Triples notation.

Chapter 4

Tutorial

4 Tutorial

In this tutorial, you will do the following:

- Create an OWL Lite ontology from scratch;
- Create an OWL DL ontology from scratch;
- Create a set of RDF resources based on the OWL DL ontology;
- Create an RDF document that uses Dublin Core vocabulary to describe metadata associated with document-type resources.

The OWL Lite and OWL DL ontologies you create are small ontologies that take you through the various features of SemanticWorks. After you have finished creating the ontologies, you will have learned how to use all the features of SemanticWorks you need to quickly construct any type of ontology. The RDF-document-creation part of the tutorial shows you how you can use the resources of an ontology to create and define RDF resources, and includes a tutorial that shows how to create an RDF document that uses the Dublin Core vocabulary.

Doing the tutorial

The OWL Lite and OWL DL parts of the tutorial start from scratch, so you do not need any file to start with and can start on these parts of the tutorial as soon as you have successfully installed SemanticWorks. The files created in these two parts of the tutorial are delivered in the `Examples` folder of your SemanticWorks application folder.

The OWL DL ontology used to create an RDF document based on an OWL DL ontology is the OWL DL ontology you will create in the OWL DL part of the tutorial It is therefore better if you go through the OWL DL part of the tutorial before you do the part in which you create an RDF document based on this ontology. However, since the required ontology is available in the `Examples` folder of your SemanticWorks application folder, you can start with the RDF document creation part if you like. The tutorial part for creating an RDF document based on the Dublin Core starts from scratch, so, again, you can start with this part right away.

It is best to do the tutorial in sequence, that is, starting with the creation of an OWL Lite ontology and ending with the creation of an RDF document that uses the Dublin Core vocabulary. The reason for this is that basic usage mechanisms and concepts are explained in detail and incrementally as you proceed in sequence. Before starting this tutorial, we also suggest that you read through the Interface section in the Introduction so as to familiarize yourself with the GUI and the terms used to describe it. Also, note that the screenshots in this tutorial draw the graphs in Detail View horizontally, from left to right; this is a setting made in the Options dialog.

About the ontologies and naming conventions

The OWL Lite ontology you will create is used to describe Altova products, while the OWL DL ontology describes Altova documents. These ontologies have deliberately been kept simple and small, so that you can concentrate on the usage mechanisms and become familiar with SemanticWorks, instead of being overwhelmed by the complexity of the ontologies.

In this tutorial we use the following naming conventions:

- Class names and instance names are capitalized (for example, `Documents`). If a class name consists of more than one word, the words are run together with each word capitalized (for example, `ProductManual`).
- Property names start with a lowercase alphabet (for example, `source`). If a property name consists of more than one word, the words are run together with words subsequent to the first capitalized (`responsibilityOf`).

 © 2006 Altova GmbH

Note about namespaces
The namespaces used for the AltovaProduct and AltovaDocument ontologies are fictitious; there are no ontologies or any other resource at the locations specified by the URIs in these namespaces.

4.1 OWL Lite Ontology

The OWL Lite ontology you will create describes Altova products. It consists of the following sections:

- Creating a New Ontology: Shows how to create a new ontology document, select an ontology level, and save the ontology document.
- Declaring Namespaces: Explores the Text View and explains what namespaces are required in your ontology document. You will declare namespaces for the AltovaProducts vocabulary and for XML Schema datatypes.
- Creating Classes: Describes how classes are created in RDF/OWL View, and explores RDF/OWL View and Text View. Explains how to delete and re-create classes, and shows how the syntax and semantics of the ontology can be checked in SemanticWorks.
- Creating the Class Hierarchy: Explains how classes are created as subclasses of another class, and thus how a hierarchy of classes can be built. Describes how the Detail View of RDF/OWL View works.
- Defining Properties: Shows how to create OWL properties—both object and datatype—and how to define the domain, range, and cardinality of properties. Also shows how relationships between classes and properties can be viewed in SemanticWorks.
- Declaring Instances: Describes how to create instances, how to define these as instances of a particular class, and how to assign a literal value to an instance. Concludes by showing how created instances can be viewed in the Classes Overview.
- Declaring AllDifferent Instances: Explains how to collect instances in a group and define each of them as being pairwise different from all other members of the group.

The OWL Lite ontology that is the end result of this tutorial part is delivered in the SemanticWorks package as the file `AltovaProducts.rdf`; it is located in the `Examples` folder in the SemanticWorks application folder.

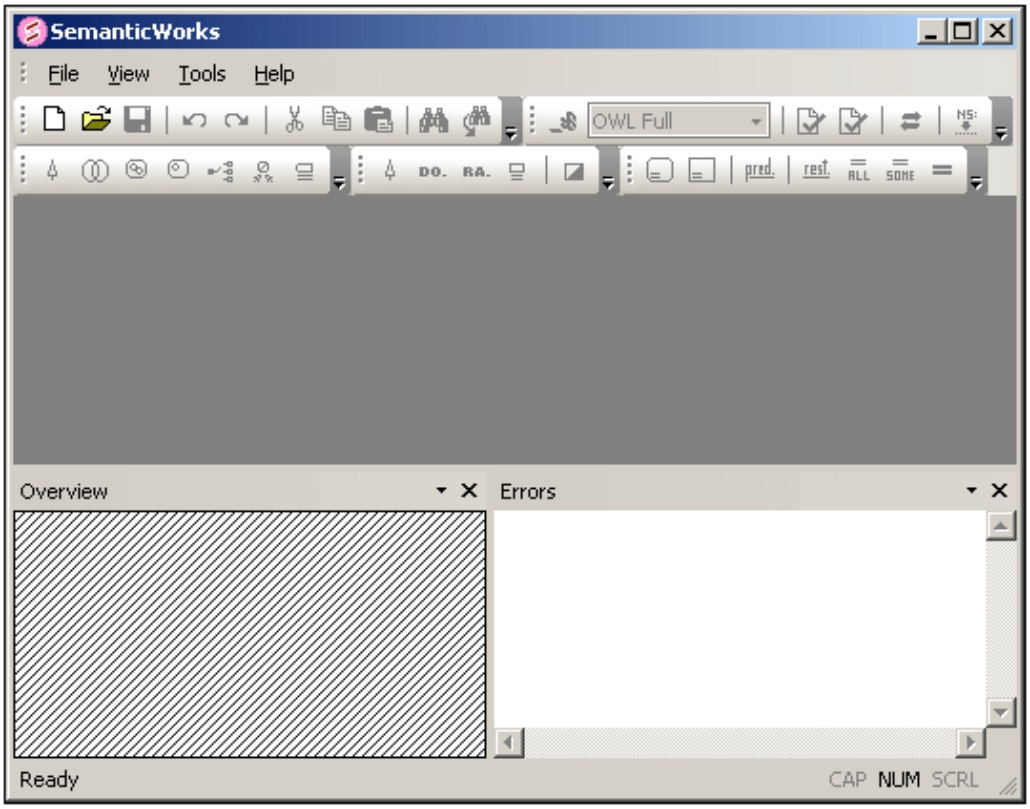
© 2006 Altova GmbH

4.1.1 Creating a New Ontology

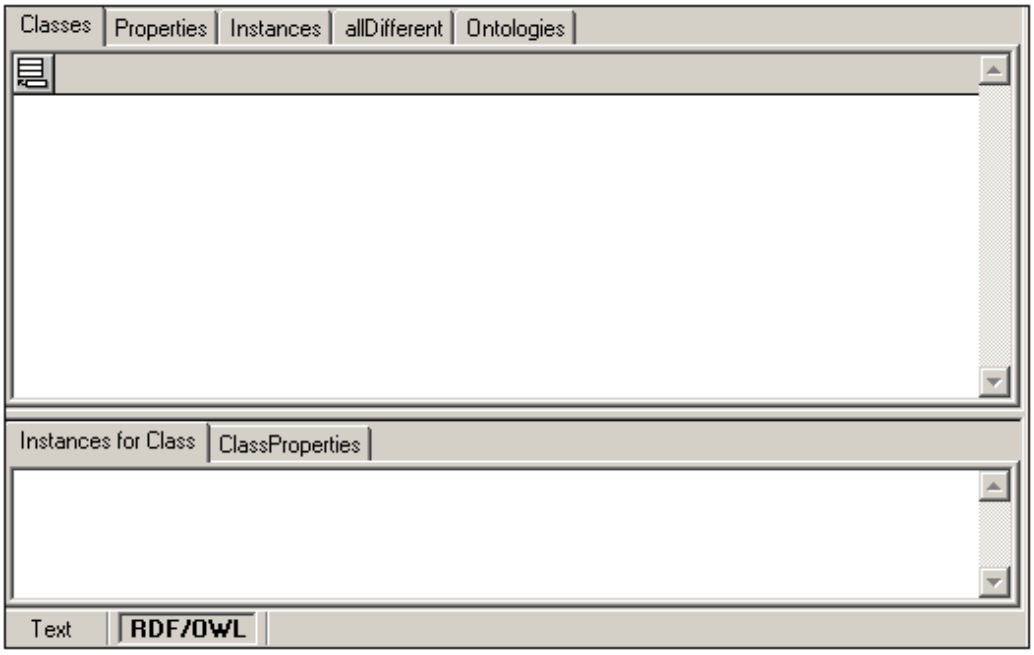
In this section, you will learn how to create a new ontology document in SemanticWorks. You will open a blank document in SemanticWorks, select an ontology compliance level (OWL Lite), and save the document, while briefly exploring the SemanticWorks interface.

Creating a new ontology document in SemanticWorks
To create a new ontology document in SemanticWorks, do the following:

1. Start SemanticWorks by clicking the SemanticWorks shortcut in the Quick Launch tray or via the **Start | All Programs** menu item. SemanticWorks starts. The application window looks something like this.



   Notice that there are three windows: (i) the Main Window; (ii) the Overview Window; and (iii) the Errors Window.
2. If the windows are not arranged as shown in the screenshot above, try arranging it in this way. Do this by dragging the title bar of the Overview Window and dropping it on the left-pointing arrow that appears in the Errors Window. The Errors Window itself should be located at the bottom of the application window (down-pointing arrow of outer circle arrows).
3. Click the New icon in the toolbar (or **File | New** or **Ctrl+N**) to open a blank document in SemanticWorks. The Main Window will now look like this.

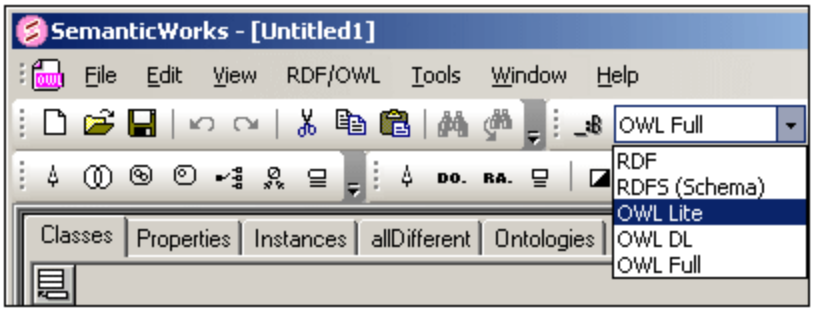


Notice that there are five tabs at the top of the window and that the Classes tab is selected. These five tabs organize the ontology information in five categories, thus providing an overview of ontology information. We refer to this view as the Ontology Overview (also called Overview for short; it should not be confused with the Overview Window). Further, notice that a pane containing subsidiary categories for the selected Overview category (currently, the Classes category) is displayed below the main category. With the Classes tab selected, the subsidiary pane displays the instances and properties of the selected class.

Selecting the language level of the ontology

The ontology you will create in this part of the tutorial will use features of the OWL Lite sublanguage. SemanticWorks checks ontologies according to the language level selected by the user, and also makes available constructors specific to the selected ontology language. It is therefore best to select the required language at the outset. To select the OWL Lite sublanguage, do the following:

1. In the RDF/OWL Level combo box in the toolbar, click the arrowhead to display the dropdown list of options (*screenshot below*). (Alternatively, select **RDF/OWL | RDF/OWL Level** to display a submenu of language levels.)



2. Select `OWL Lite` from the dropdown list.

Notice that as soon as the specification level is selected, a syntax check is run on the document, and the message `This ontology is well-formed` is displayed in the Errors

Window.

Note: You can change the RDF/OWL level at any time. The selected level is retained through changes of views (Text View and RDF/OWL View). When you reopen an ontology document, you should check that the desired language level is selected.

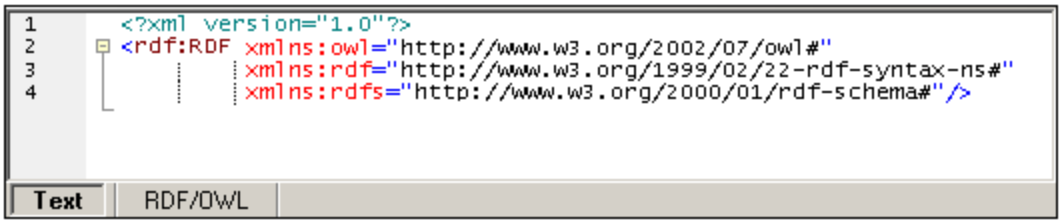
Saving the ontology
It is a good idea to save the ontology before continuing. Save the file using any name you like and with either the `.rdf` or `.owl` file extension. (The `.rdf` extension is allowed because OWL ontologies are themselves RDF documents; it is, in fact, more usual than the `.owl` extension.) We'll assume that the file you create in this part of the tutorial is called `AltovaProducts.rdf`.

Having done all of the above, you are now ready to declare namespaces for your ontology.

4.1.2 Declaring Namespaces

In this part of the tutorial, you are creating an OWL Lite ontology for Altova products using a custom-made vocabulary which belongs to a unique namespace. Before starting to use this vocabulary in the ontology, you must, in the interests of good practice, declare the namespace for the vocabulary, as well as any other namespaces you require. Declaring the required namespaces is what you will do in this section. In the course of carrying out these steps, you will also see the RDF/XML format of your ontology in Text View.

Text View of SemanticWorks
To see how the RDF/XML representation of the newly created ontology looks, click the Text View button at the bottom of the Main Window. The Text View will look something like this.

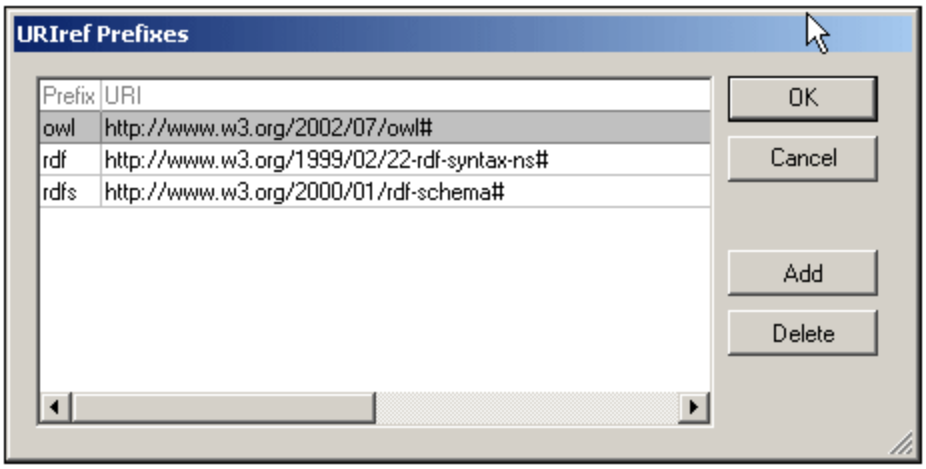


Notice that there is a single element `rdf:RDF`, which is the root element of every OWL ontology. It has three namespace declarations (for the RDF, RDFS, and OWL namespaces). SemanticWorks inserts the RDF, RDFS, and OWL namespaces by default; they are required so as to be able to use RDF, RDFS, and OWL elements and attributes in the ontology document.

Note: You can configure Text View fonts in the Options dialog (**Tools | Options**).

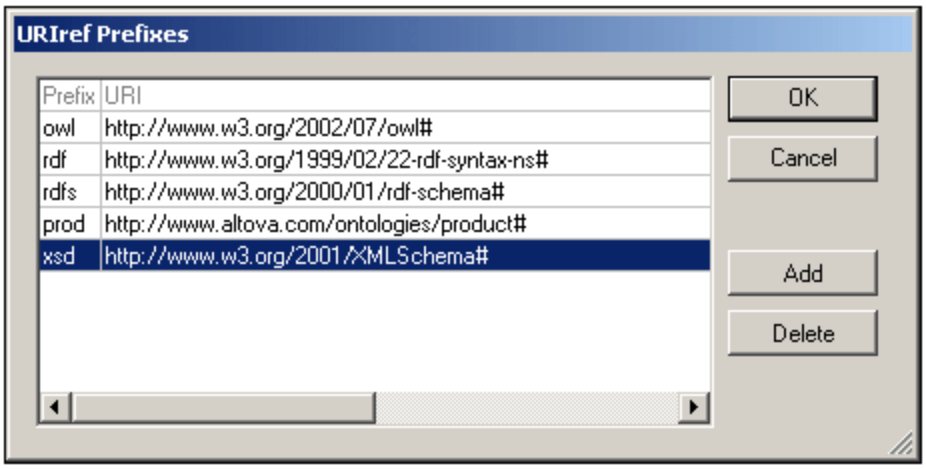
Declaring namespaces for the ontology document
Your Altova product vocabulary will use the namespace `http://www.altova.com/ontologies/product#`, and this needs to be declared. You will also need to declare the XML Schema namespace so you can use the XML Schema datatypes for OWL datatype properties. Declare namespaces as follows:

1. Select RDF/OWL View.
2. Select the command **Tools | URIref Prefixes**. This pops up the URIref Prefixes dialog.



3. In this dialog, click the Add button to add a line for a new namespace declaration.
4. In the Prefix column, enter `prod`. In the URI column, enter `http://www.altova.com/ontologies/product#`.

5. Next, add the XML Schema namespace `http://www.w3.org/2001/XMLSchema#`, giving it a prefix of `xsd`. The URIref Prefixes dialog will look something like this:



5. Switch to Text View to check that the newly declared namespaces have been correctly added.

You have now declared all the namespaces for your ontology, and you can now start creating ontology items.

Note:

- It is important to declare namespaces at the very outset. This ensures that URIref prefixes used in the names of ontology items are correctly expanded to the relevant namespaces when a new ontology item is created. If a namespace has not been declared, then the URIref prefix associated with that namespace, when used in the name of an item, will not be expanded to that namespace.
- The expansion of a URIref prefix to a namespace will not take place even if: (i) the relevant namespace is declared subsequent to the creation of an item; or (ii) if the ontology item is renamed after the namespace is declared but was created before the namespace was declared.
- Ontology items with unexpanded prefixes might not be correctly recognized.

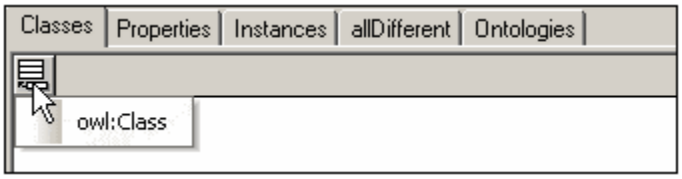
4.1.3 Creating Classes

In this tutorial, we will build the ontology using a generally top-down approach. Whatever approach one chooses, it is usual to start by defining classes. In this section you will learn how to create three classes in RDF/OWL View: `Product`, `XMLSpy`, and `Edition`, and check the ontology for correct syntax and semantics.

Creating new classes

SemanticWorks enables you to name classes so that they are represented in the RDF/XML serialization using either (i) namespace prefixes to stand for the namespace URI, or (ii) the full URIref (that is, with the prefix expanded). You will create the classes `Product` and `XMLSpy` using a different serialization method for each. Do this as follows:

1. In the Classes tab of RDF/OWL View, click the Add New button and select `owl:Class` .



This creates a line for the class in the Classes Overview.

2. With `urn:Unnamed-1` in the newly created class entry line highlighted, type in `prod:Product`, and press Enter.

This creates a class with a URIref of `http://www.altova.com/ontologies/product#Product`. Switch to Text View to see how the class has been serialized in RDF/XML. You will see something like this:

```
<rdf:Description rdf:about="
http://www.altova.com/ontologies/product#Product">
    <rdf:type>
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Class
"/>
    </rdf:type>
</rdf:Description>
```

Notice that the class name has been serialized as an expanded URIref. This is because the Expand URIref Prefixes option (**Tools | Expand URIref Prefixes** in RDF/OWL View) has been toggled on. Notice also that, although the URIref has been expanded in its serialized form, it is shown in RDF/OWL View as `prod:Product`, that is, with a prefix for the namespace part of the URIref. In RDF/OWL View, URIrefs are shown in the form in which they are entered.

3. In RDF/OWL View, create two new classes and name them `prod:XMLSpy` and `prod:Edition`. In Text View, you will see that the two new classes have been serialized as expanded URIrefs. In RDF/OWL View, the URIrefs are shown in the form in which they were entered (that is, `prod:XMLSpy` and `prod:Edition`).

 © 2006 Altova GmbH

URIrefs with Prefixes, and How to Delete a Class

If you wish to serialize the RDF/XML using URIref prefixes instead of expanded URIrefs, click the Expand URIref Prefixes toolbar icon so as to deselect it. Identifiers created from now onwards (and till this option is toggled on again) will be serialized with names in the form `prefix:localname`.

From now on, every input is solely understood as a URI (absolute or relative) and is **not** understood as prefix-and-local-name. That is, prefixes that would have been expanded to a URI part when the option is activated are now simply used as the scheme part of the URI that is entered. Note that all relative URIs will be resolved against the document's global base URI. Furthermore, note that choices offered by dropdown boxes are also affected by this option; so if the setting is that URIref prefixes are not expanded, dropdown boxes always show full URIs instead of abbreviated URIrefs.

To see how unexpanded URIrefs are serialized, do the following:

1. Delete the class `XMLSpy` (by selecting it and clicking **Edit | Delete**).
2. Deselect Expand URIref Prefixes so it is toggled off.
3. Re-create the class `XMLSpy`, naming it `prod:XMLSpy`.

The Text View will show the class serialized as `prod:XMLSpy` with the prefix unexpanded.

If you do try this out, be sure to once again delete the `XMLSpy` class and re-create it as it was originally created, that is with URIref prefixes expanded.

Checking the ontology

You now have a simple ontology that declares three URIrefs to be three unrelated OWL classes. To check the syntax of the ontology, click the Syntax Check icon in the toolbar (**RDF/OWL** menu). The message `This ontology is well-formed` appears in the Errors Window. Now check the semantics of the ontology by clicking the Semantics Check icon in the toolbar. The message `This ontology is at least partially consistent` appears in the Errors Window. This is a valid ontology (has correct syntax and is partially consistent), but does not provide much information about the three classes. (For more information on consistency evaluation, see Semantics Check.)

In the next sections of the tutorial, you will create the class hierarchy, which will link classes semantically with each other.

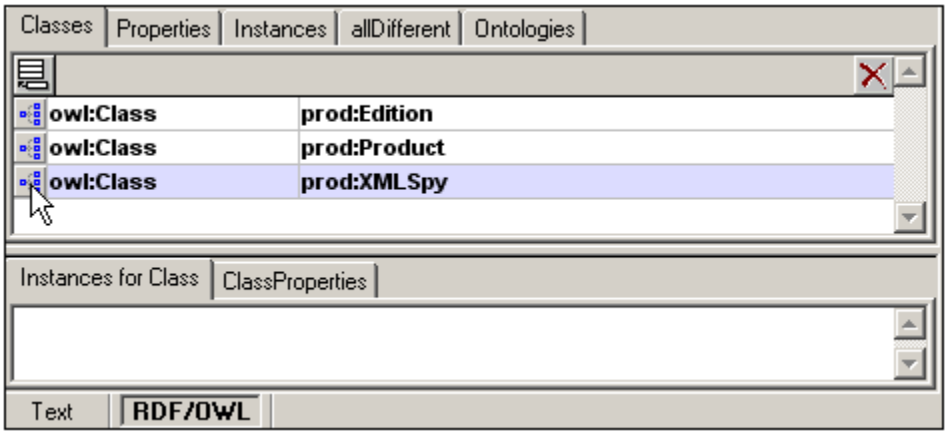
4.1.4 Creating the Class Hierarchy

In this section, you will learn how the three classes (`Product`, `XMLSpy`, and `Edition`) created in the previous section can be linked in a simple hierarchy. What we wish to do is:

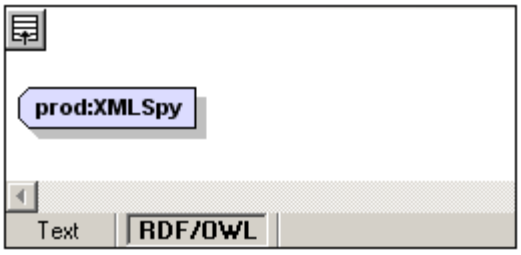
- Define `XMLSpy` as a subclass of `Product`, which essentially states that any instance of the `XMLSpy` class must also be an instance of the `Product` class.
- Use the `Edition` class to (i) define it as the range of a property called `prod:hasEdition`, and (ii) create instances of `Edition`.

Creating a class as a subclass of another
To create the class `XMLSpy` as a subclass of the class `Product`, do the following:

1. In the Classes tab of RDF/OWL View, click the Detail View button of the `prod:XMLSpy` class (*screenshot below*).

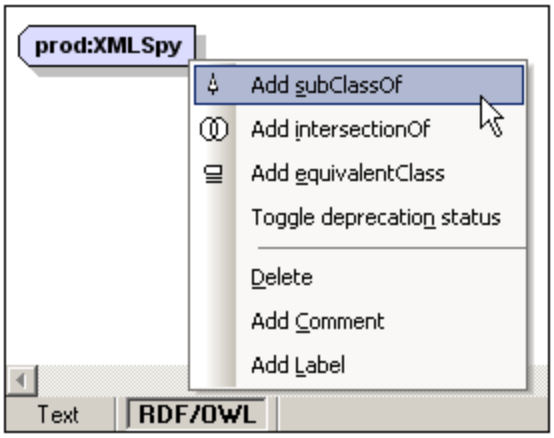


This switches the view to the Detail View of the `XMLSpy` class (*screenshot below*). Notice the shape of the classes box. All classes in Detail View are indicated by this arbitrary-hexagon-shaped box.

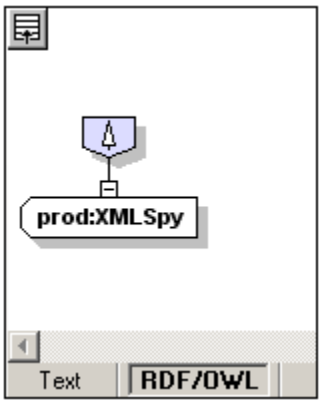


2. In Detail View, right-click the `prod:XMLSpy` classes box. This pops up a context menu (*screenshot below*).

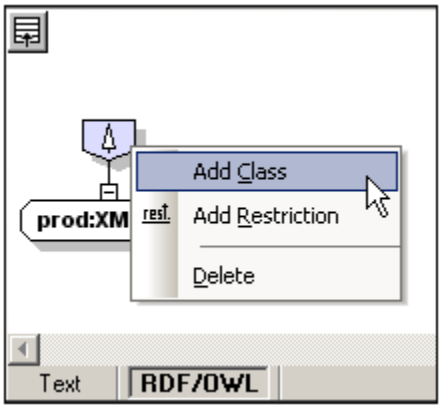
© 2006 Altova GmbH



3. Select `Add subClassOf` from the context menu. This adds a `subClassOf` connector to the `prod:XMLSpy` box (*screenshot below*).

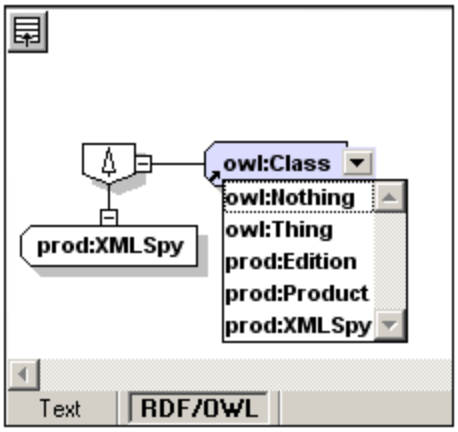


4. Right-click the `subClassOf` connector, and, in the context menu that appears, select `Add Class` (*screenshot below*).



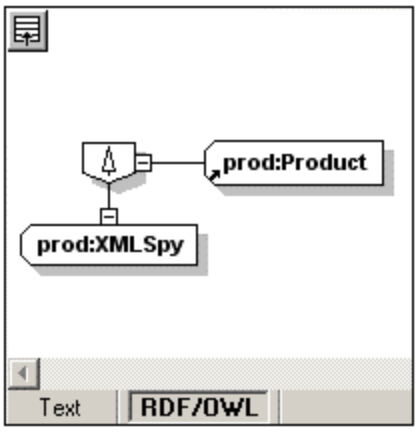
A class box is added that is linked to the `subClassOf` connector (*screenshot below*), indicating that the class represented by this class box is a subclass of the `XMLSpy` class.

5. To select which class this class box represents, click the downward-pointing arrow at the right-hand side of the class box. This drops down a list of available classes (*screenshot below*).



Notice that the set of available classes consists of the classes you have declared in this ontology plus the two general OWL classes `Thing` and `Nothing`.

6. Select `prod:Product` from the dropdown list to complete the definition (*screenshot below*).



The diagonal arrow at bottom left of the `Product` class box indicates that the box is a reference to the class `Product`.

Having carried out the steps above, you have defined that the class `XMLSpy` is a subclass of the class `Product`. Now do the following:

- Check the Text View to see the RDF/XML serialization of your new definition.
- Check the syntax and semantics of the modified ontology. Both syntax and semantics checks should return positive results.

Relating a class to its properties and instances

In the steps above, you have learned how to define relationships between two classes. In SemanticWorks, relationships between classes are defined in the Detail View of the appropriate class. To define a relationship between a class and its property (for example, the domain and range of a property), the definition is made on the property. Similarly, the definition that an instance is an instance of a particular class is made on the instance.

In this tutorial, we wish to do the following:

- Define the class `XMLSpy` to be the domain of the property `hasEdition`, and the class `Edition` to be the range of the property `hasEdition`. This would mean that the property `hasEdition` applies to the class `XMLSpy` and takes values that are instance

of the class `Edition`.
- Declare instances of the class `Edition`.

These property definitions and instance declarations are made in the following sections.

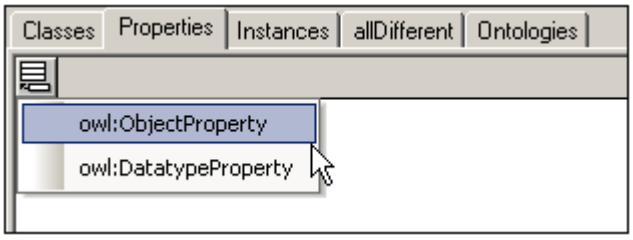
4.1.5 Defining Properties

Properties are created at a global level and then related to different classes. In our ontology, we require two properties:

- `hasEdition` to carry information about the edition of a product. The edition can be Enterprise, Professional, or Home. We will create this property as an *object property*. Doing this enables us to relate one resource to another. In this case we wish to relate instances of the `XMLSpy` class to instances of the `Edition` class via the `hasEdition` property. The class (or classes) that the property applies to is called the property's domain, while the set of values the property can take is called the property's range.
- `version`, which is a literal value indicating the year in which a product is released. We will create this property as a *datatype property*. It relates instances of the `XMLSpy` class to a positive integer (which is the year of release and gives the version of the product).

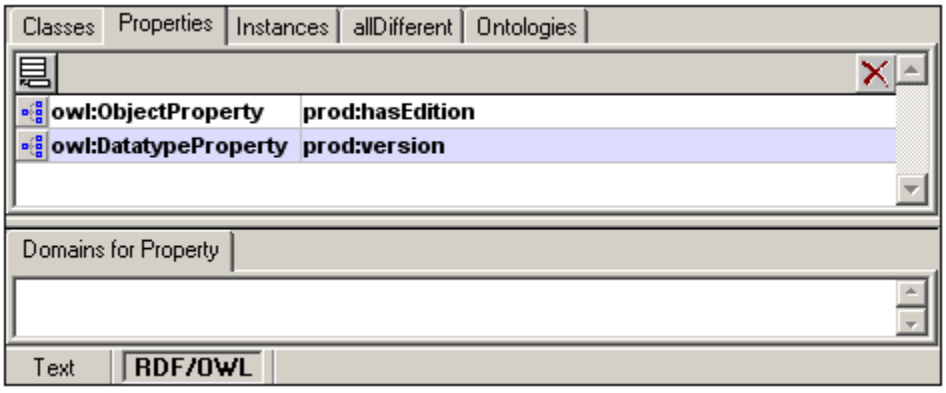
Creating properties
Properties are created in the same way that classes are created, by clicking the Add New button and then specifying the name of the property to be created. To create a property, do the following:

1. In the Properties tab of RDF/OWL View, click the Add New button, and select `owl:ObjectProperty`.



2. This creates a line for the newly created object property. In this line, enter `prod:hasEdition` as the name of the new object property.
3. Now add a datatype property by: (i) clicking the Add New button and selecting `owl:DatatypeProperty`, and (ii) entering `prod:version` as the name of the datatype property.

You have now created two properties: (i) an object property called `hasEdition`, and (ii) a datatype property called `version`. The Properties tab should now look like this:



You are now ready to define the properties in Detail View.

 © 2006 Altova GmbH

Defining properties

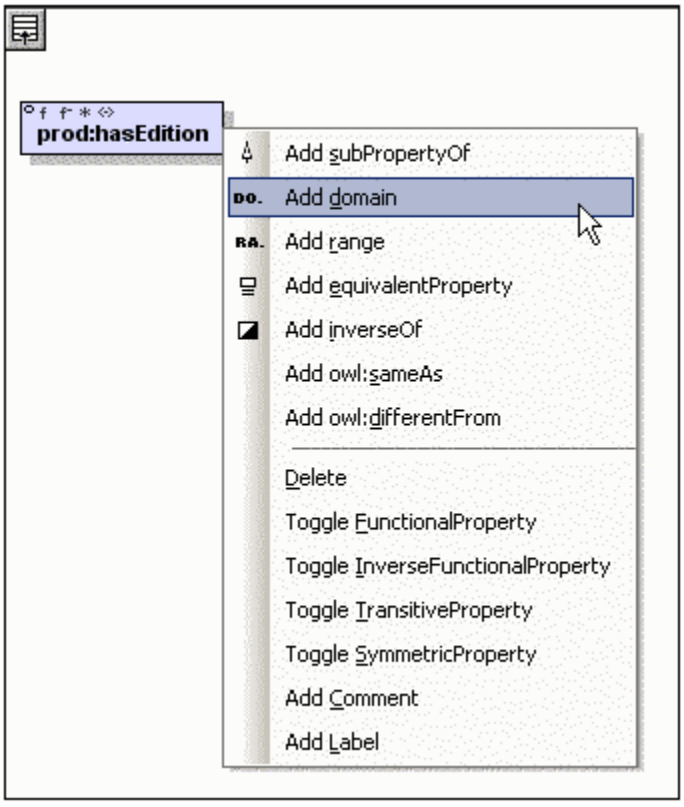
The first thing you should note when working with OWL properties in SemanticWorks is that object properties and datatype properties are indicated with slightly different symbols in Detail View. Double-click the Detail View icon to see the Detail View symbol for each property:

Object properties are indicated with an O icon in the top-left corner, datatype properties are indicated with a DT icon. The icons to the right of these two icons are actually toggle switches for specifying the cardinality constraints and characteristics of properties. They are toggles for, from left to right, setting the property to be a functional property; an inverse-functional property; a transitive property; and a symmetric property. Note that a datatype property is only allowed the functional property relationship.

Defining an object property

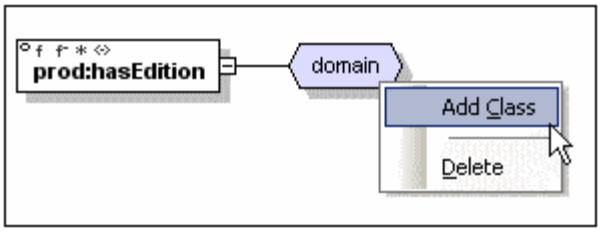
You will now define the relationships and characteristics of the two properties you have created. Do this as follows:

1. Double-click the Detail View icon of the `hasEdition` property. This brings up the Detail View of the property.
2. Right-click the `prod:hasEdition` box, and, from the context menu that appears, select Add Domain (*screenshot below*).



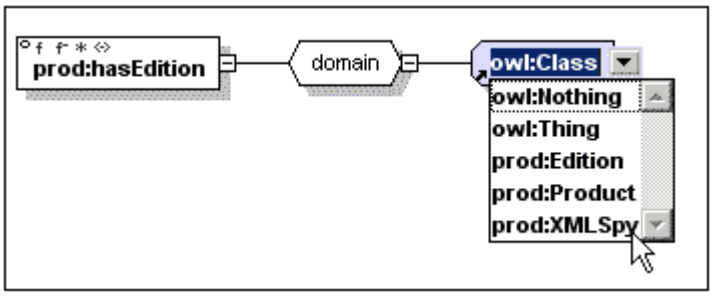
The Domain connector box is inserted.

3. Right-click the Domain connector box, and select Add Class (*screenshot below*).



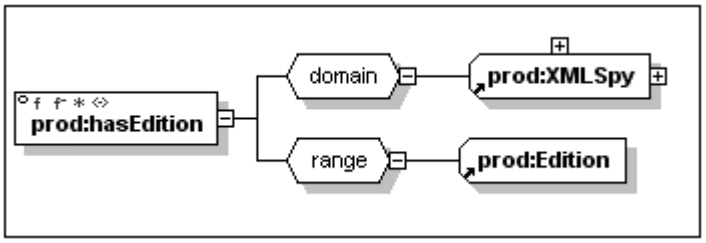
A Class box is inserted.

4. Click the down arrow in the Class box, to drop down a list of available classes, and select `prod:XMLSpy` (*screenshot below*).



The class `prod:XMLSpy` is set as the domain of the property `prod:hasEdition`. This relationship states that the `hasEdition` property is applicable to the class `XMLSpy`.

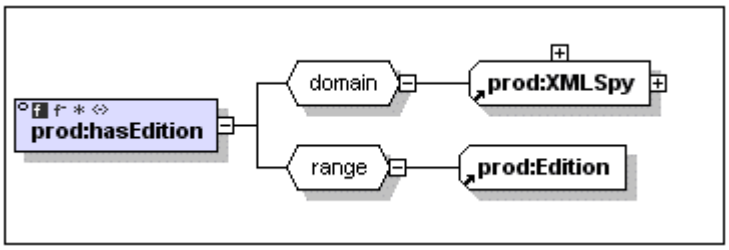
5. In the same way that you set the domain of the property, now set the range of the property: (i) right-click the `hasEdition` property box; (ii) select Add Range; (iii) right-click the Range connector box; (iv) select Add Class; (v) from the dropdown list in the Class box, select the class `prod:Edition`. The Detail View of the `hasEdition` property should look like this:



The range of the property is now set to be instances of the class `Edition`.

Defining the cardinality constraint of a property

We wish to specify that the property `hasEdition` can take only one instance of the `Edition` class as its value. This is done by specifying that the `hasEdition` property is functional. To make the property functional, click the `f` icon (functional property icon) in the `hasEdition` property box. The Detail View of the `hasEdition` property should look like this:

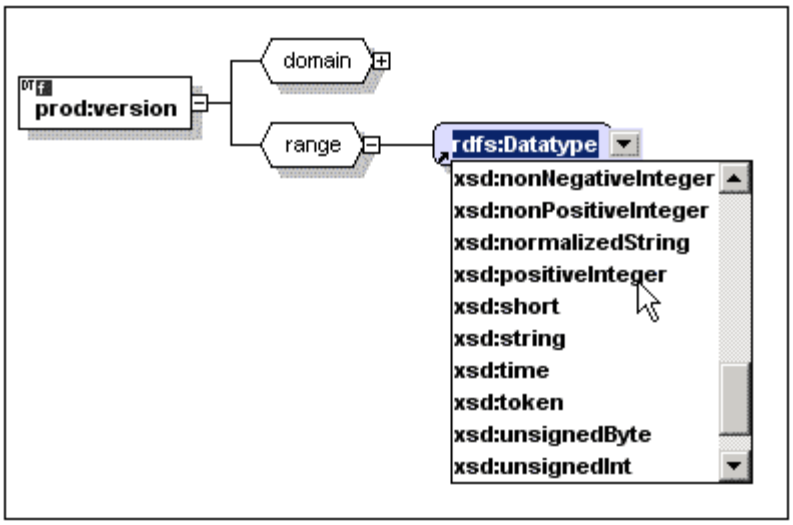


Notice that the `f` icon in the `hasEdition` property box is highlighted, indicating it is toggled on.

 © 2006 Altova GmbH

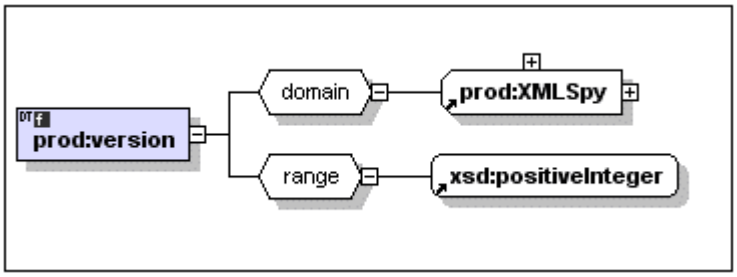
Defining a datatype property
For the datatype property `prod:version`, you will define a domain similarly as defined above for the object property `has:Edition`. Set the domain to the class `XMLSpy`. For the range, we wish to define the XML Schema datatype `xsd:positiveInteger`. Do this as follows:

1. Add the Range connector box, by right-clicking the `version` property box and selecting Add Range.
2. Right-click the Range connector box, and select Add XML Schema Datatype. A Datatype box is added.
3. From the dropdown list in the Datatype box, select `xsd:positiveInteger` (*screenshot below*).

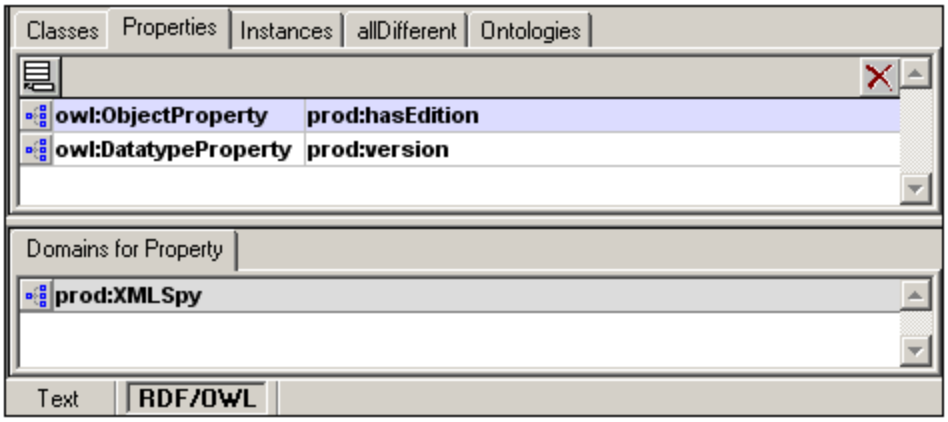


4. Set the cardinality of the `version` property by toggling on the functional property specification.

The `version` property is now a functional property. It is applicable to instance of the class `XMLSpy` (its domain) and can take values that are of the XML Schema datatype `xsd:positiveInteger` (its range). The Detail View of the `version` property should look like this:

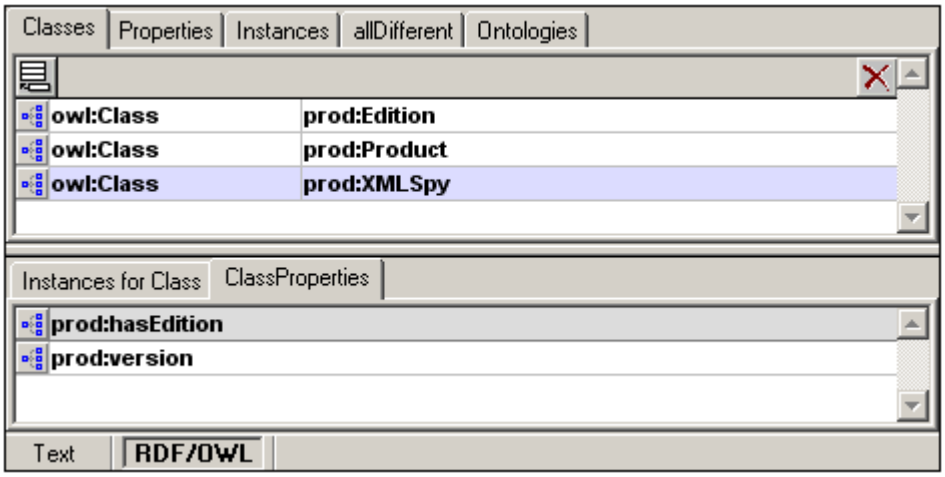


Domain listing in Properties Overview
When you switch to the Properties Overview (that is, the Properties tab in the Ontology Overview), notice that the domains of the selected property are displayed in the subsidiary window (*screenshot below*).



Click the Detail View button of a domain class to go directly to the Detail View of that class.

Class properties in Classes Overview
You can see the properties of a selected class in the subsidiary Class Properties window of the Classes Overview (Classes tab in the Ontology Overview). *See screenshot below*.

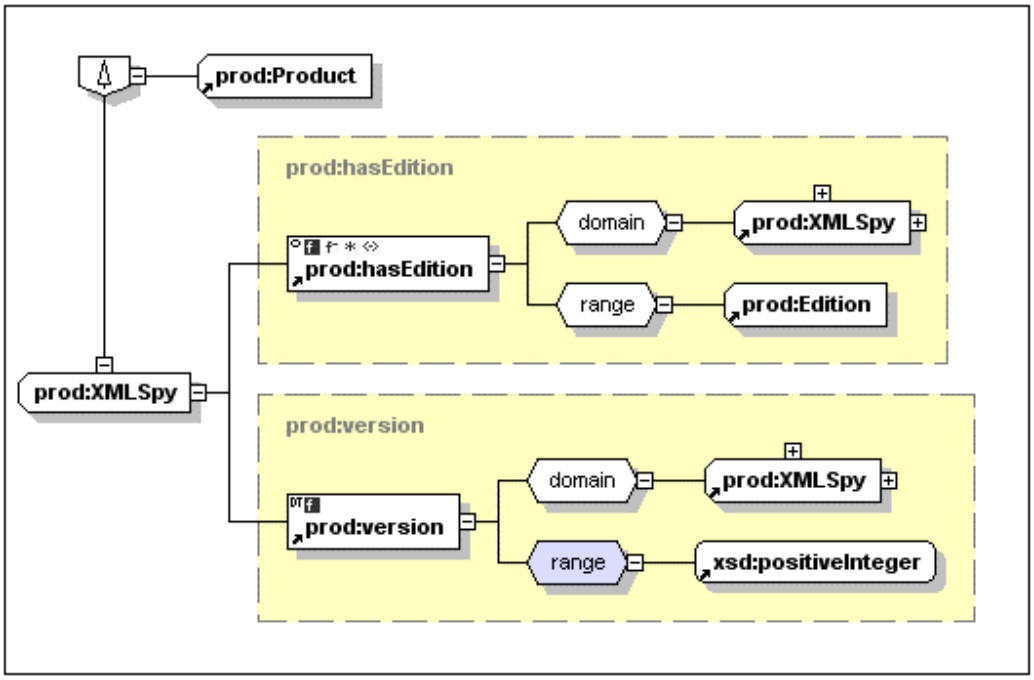


Click the Detail View button of a property to go directly to the Detail View of that property.

Checking the syntax and semantics of the ontology
At this stage, check the syntax (**RDF/OWL | Syntax Check**) and the semantics (**RDF/OWL | Semantics Check**) of your ontology, making sure that the OWL Lite level is selected. Your OWL Lite ontology till this point should be both well-formed and at least partially consistent.

Checking the relationships between properties and classes
Both properties (`hasEdition` and `version`) were set to have the class `XMLSpy` as its domain, meaning that both properties are applicable to the `XMLSpy` class. Now check the effect of these definitions on the `XMLSpy` class. Do this as follows:

1. Go to the Document Overview (by clicking the Overview icon located at the top left of Detail View).
2. Select the Classes tab.
3. Go to the Detail View of the class `XMLSpy` (by clicking the Detail View icon next to the class entry). The Detail View of the XMLSpy class should look something like this:



Here we see that the class `XMLSpy` is a subclass of the class `Product`, and has two properties: the object property `hasEdition` and the datatype property `version`.

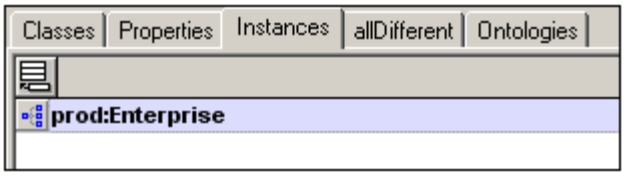
4.1.6 Declaring Instances

So far you have created three classes, `Product`, `XMLSpy`, and `Edition`, and two properties, the object property `hasEdition` and the datatype property `version`. You have defined both properties to apply to the `XMLSpy` class (by making this class the domain of the properties). Further, you have defined (i) the range of the `hasEdition` property (that is the values this property can take) to be instances of the `Edition` class, and (ii) the range of the `version` property to be a literal value of the XML Schema datatype `positiveInteger`.

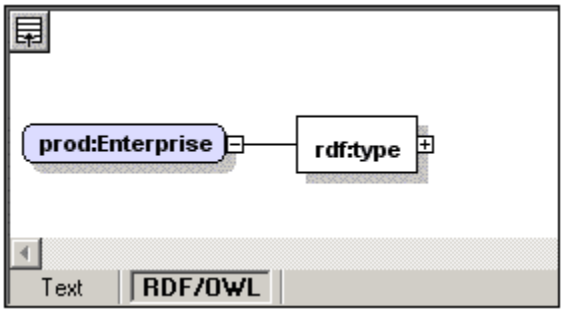
In this section, you will first create three instances of the `Edition` class, which will be simple instances, and then three more complex instances of the `XMLSpy` class.

Creating simple instances
To create an instance of the `Edition` class, do the following:

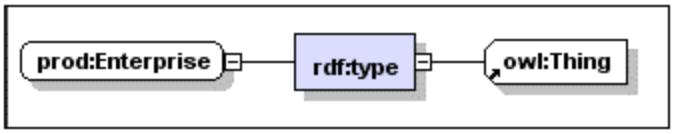
1. In the Instances tab of RDF/OWL View, click the Add New button and select Instance. This creates an entry for an instance.
2. Enter `prod:Enterprise` as the name of the instance (*screenshot below*).



3. Click the Detail View button of the `prod:Enterprise` entry to switch to Detail View, which will look something like this:

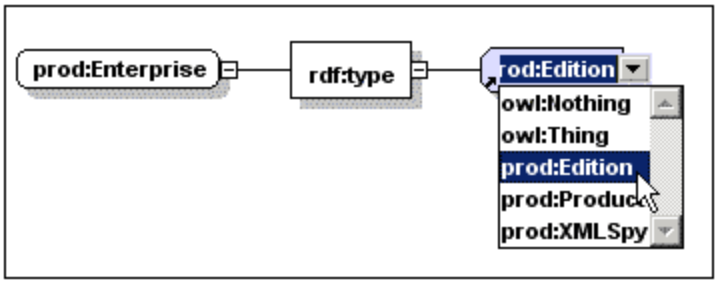


4. Expand the `rdf:type` connector by clicking the plus symbol on its right-hand side. Detail View will now look something like this:



5. Double-click the `owl:Thing` box, click the down arrow to drop down the list of available classes of which `prod:Enterprise` can be made an instance, and select `prod:Edition` (*screenshot below*).

 © 2006 Altova GmbH



You have created `prod:Enterprise` as an instance of `prod:Edition`.

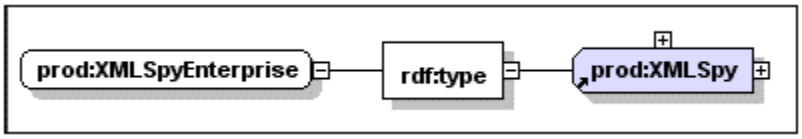
6. Check the syntax and semantics of your ontology. You should get messages saying the ontology is both well-formed and partially consistent.
7. Create two more instances of the `Edition` class, as described above, and call them `prod:Professional` and `prod:Home`, respectively.
8. Check that your ontology is well-formed and partially consistent, which it should be if you have done everything as described above.

You now have three instances of the `Edition` class: `Enterprise`, `Professional`, and `Home`.

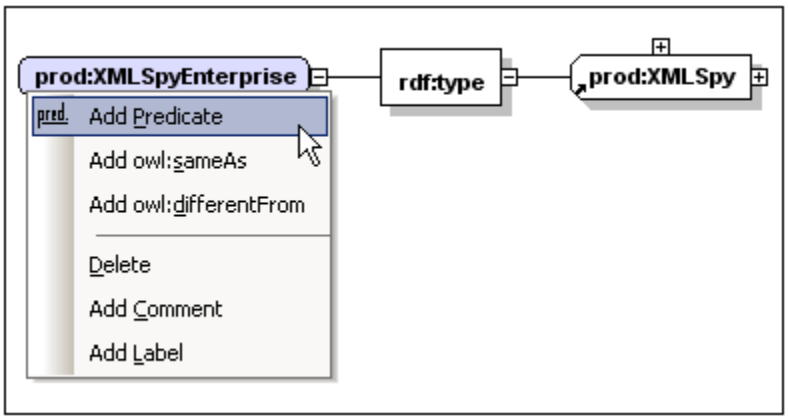
Creating instances that have predicates (properties)

You will now create three instances of the `XMLSpy` class. These instances will each additionally be defined with the two properties, `hasEdition` and `version`, that apply to the `XMLSpy` class. Create these instances as follows:

1. In the Instances tab of RDF/OWL View, click the Add New button and select Instance. Name the instance `prod:XMLSpyEnterprise`.
2. Click the Detail View button of the `prod:XMLSpyEnterprise` entry to switch to Detail View.
3. Expand the `rdf:type` connector and select `prod:XMLSpy`. This defines `XMLSpyEnterprise` as an instance of the class `XMLSpy`. The Detail View will now look something like this:

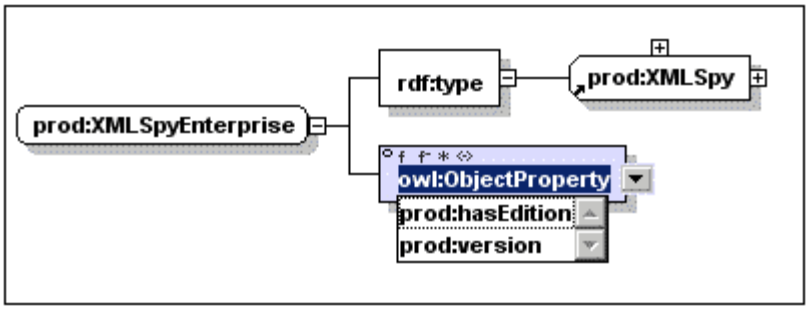


4. Right-click the `XMLSpyEnterprise` instance box, and select Add Predicate (*screenshot below*).

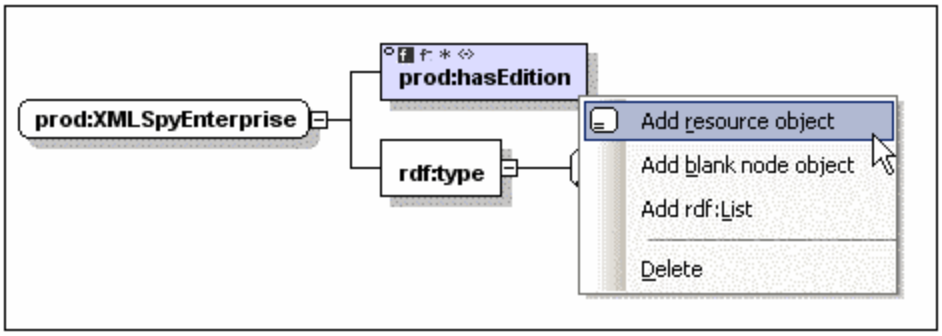


5. In the property box, click the down arrow to drop down a list of properties defined for

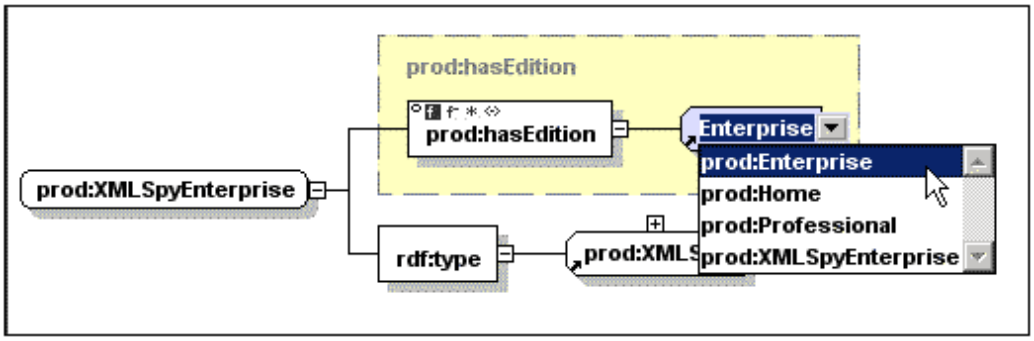
this class, and select `prod:hasEdition` (*screenshot below*).



6. Right-click the `prod:hasEdition` property box and select Add Resource Object (*screenshot below*).



7. Click the down arrow of the Resource Object box and click `prod:Enterprise` from the dropdown list of available instances (*screenshot below*).



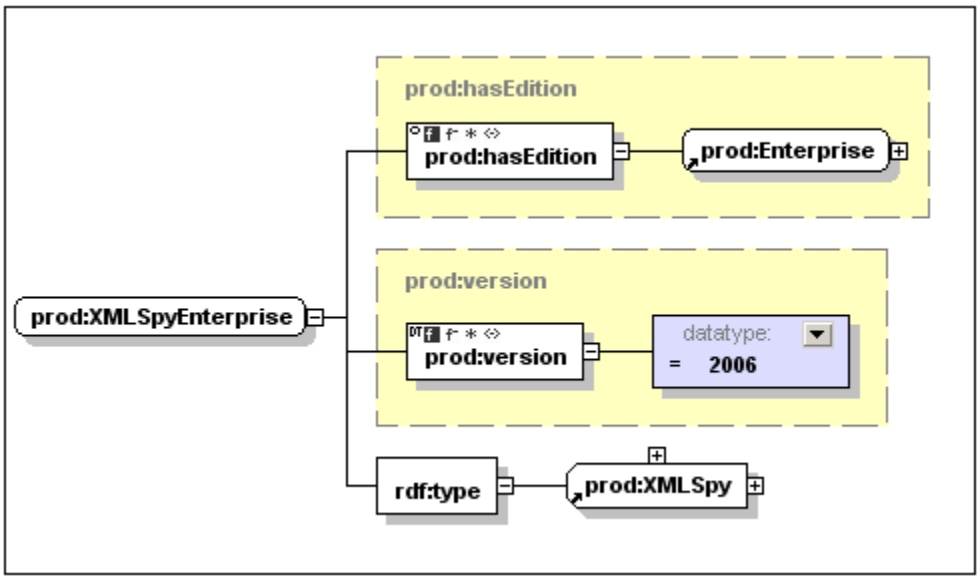
This defines that the object property `hasEdition` has the instance `Enterprise` (of the class `Edition`) as its object. Recall that you have set the range of the `hasEdition` property to be instances of the class `Edition`. If you check the semantics of the ontology, you will see that the ontology is valid (well-formed and partially consistent).

8. Double-click the `Enterprise` Resource Object box, select `XMLSpyEnterprise`, and do a semantic check. You will receive an error message saying that the resource object `XMLSpyEnterprise` is not within the range defined for the property `hasEdition`. Double-click the `Enterprise` Resource Object box, and select `Enterprise` again, which is what we want.
9. Right-click the `XMLSpyEnterprise` instance box, and select Add Predicate to add a second property.
10. Select the property `prod:version` from the dropdown list to add this property as a predicate.
11. Right-click and select Add Literal Object. (Recall that the `version` property has a literal value of the XML Schema datatype `positiveInteger` defined as its range.

 © 2006 Altova GmbH

SemanticWorks automatically provides the correct entry helper.)

12. Enter `2006` at the blinking cursor in the Literal Object box, and click Enter. The Detail View should look something like this:



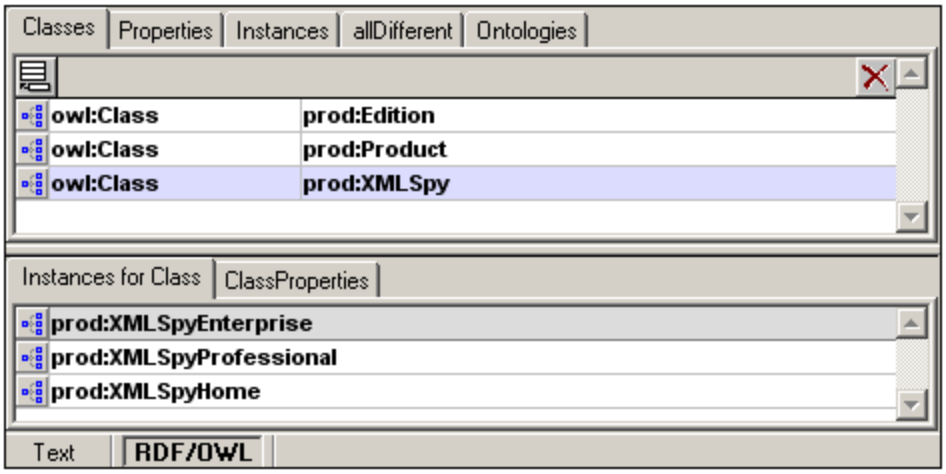
The instance `XMLSpyEnterprise` has therefore been defined to:

- Be an instance of the class `XMLSpy`,
- Have an object property `hasEdition` that takes the instance `Enterprise` as its object, and
- Have a datatype property `version` that takes the positiveInteger value `2006` as its literal value.

If you check the semantics of the ontology, you will see a message saying that the ontology is both well-formed and partially consistent. Now complete the ontology by creating two more instances of the `XMLSpy` class. Call them `XMLSpyProfessional` and `XMLSpyHome`, respectively. Define them just as you defined the `XMLSpyEnterprise` instance, with the only difference being that they should have the `Professional` and `Home` instances, respectively, as the objects of the `hasEdition` property.

Class instances in Classes Overview

You can see the instances of a selected class in the subsidiary Instances for Class window of the Classes Overview (Classes tab in the Ontology Overview). In the screenshot below, the class `XMLSpy` is selected. The Instances for Class subsidiary window shows the instances for the `XMLSpy` class.



Click the Detail View button of an instance to go directly to the Detail View of that instance.

© 2006 Altova GmbH

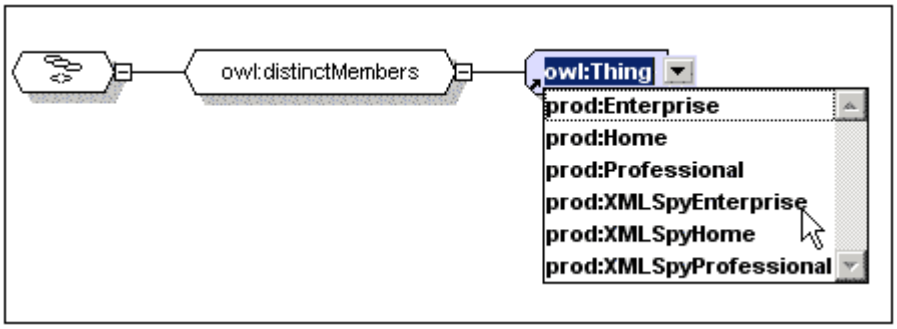
4.1.7 Declaring AllDifferent Instances

In the previous section, you created three instances of the `XMLSpy` class: `XMLSpyEnterprise`, `XMLSpyProfessional`, and `XMLSpyHome`. These are the three editions of the XMLSpy product. You specified the difference in the editions by using the property `hasEdition` and assigning it three different object values, namely the instances `Enterprise`, `Professional`, and `Home`, respectively. This, however, does not ensure that the three URIrefs `prod:XMLSpyEnterprise`, `prod:XMLSpyProfessional`, and `prod:XMLSpyHome` are indeed different, since two of them, or even all three, could actually be referring to a single individual (or resource). That these are three different URIrefs must be explicitly stated, and the AllDifferent construct is used to state the pairwise difference of instances in a collection.

In this section of the tutorial, you create an AllDifferent collection object and collect within it all the instances that must be pairwise different. Do this as follows:

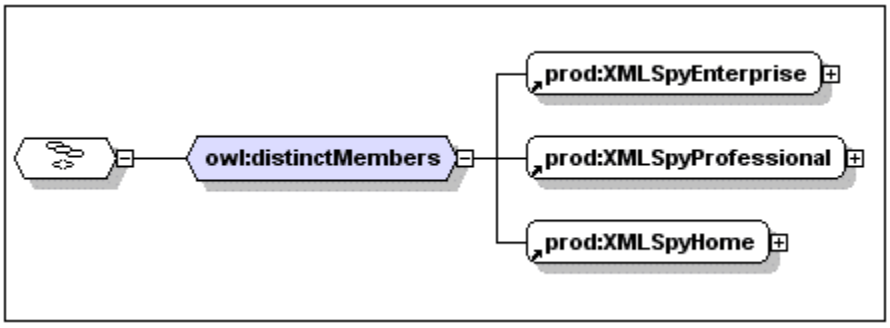
1. In the allDifferent tab of RDF/OWL View, click the Add New button and select allDifferent. Name the newly created allDifferent item `prod:XMLSpyEditions`.
2. Click the Detail View button of the `prod:XMLSpyEditions` entry to switch to Detail View (*screenshot below*).

3. Right-click the `owl:DistinctMembers` box, and select Add Instance.
4. Click the down arrow in the newly created instance box, and select `prod:XMLSpyEnterprise` (*screenshot below*).



The `XMLSpyEnterprise` instance is selected as a distinct member of the `XMLSpyEditions` AllDifferent collection.

5. Add the `XMLSpyProfessional` and `XMLSpyHome` instances to the collection (by right-clicking the `owl:DistinctMembers` box and selecting Add Instance, then selecting the respective instances). When you're done, the Detail View will look something like this:



6. Run a semantic check to confirm the validity (correct syntax and partial consistency) of

the ontology, and then save the file.

That's it!
You have successfully completed the OWL Lite tutorial. In the course of this tutorial you have learned how to build an OWL Lite ontology using SemanticWorks and have become familiar with the interface, features, and mechanisms of SemanticWorks.

© 2006 Altova GmbH

4.2 OWL DL Ontology

The OWL DL ontology you will create describes Altova documents. It shows you how to use SemanticWorks features that are not available when editing OWL Lite ontologies. It consists of the following sections:

- Setting up an OWL DL Ontology: Shows how to create a new ontology document, select the ontology level, declare namespaces, and save the ontology document.
- Creating Classes and Hierarchies of Classes: Shows how to create the ontology classes and create a hierarchy of classes using the subclass connector.
- Instances as Class Enumerations: Explains how instances can be created as class enumerations, and why class enumerations are needed.
- Defining Properties: Shows how to create OWL object and datatype properties, and how to define their domain and range.
- Describing Classes and Their Instances: The focus is on describing complex OWL DL relationships between classes. You ill learn how to describe a class as a union of two classes and to further restrict that union. You conclude by creating instances of the newly created restricted class and checking the syntax and semantics of the ontology.
- Defining Complementary Classes and Their Instances: Explains how to state that one class is a complement of another class. Class instaces are then created and defined, and the ontology is checked for correct syntax and semantics.

The OWL DL ontology that is the end result of this tutorial part is delivered in the SemanticWorks package as the file `AltovaDocuments.rdf`, and it is located in the `Examples` folder in the SemanticWorks application folder. Note that the `AltovaDocuments.rdf` ontology is required for the RDF tutorial part, which uses its resources.

Note: In this part of the tutorial we assume that you have already completed the earlier OWL Lite part of the tutorial. Therefore, certain steps that were explained in detail in the earlier part are not described in detail in this part. If you find that you are having difficulty with any step, refer back to the corresponding section in the first (OWL Lite) part of the tutorial.

4.2.1 Setting Up an OWL DL Ontology

Setting up an OWL DL ontology in SemanticWorks involves the same steps as with an OWL Lite ontology. These are:

1. Creating a new ontology document.
2. Setting the language level (OWL DL in this case).
3. Declaring namespaces.
4. Saving the file with a `.rdf` or `.owl` extension.

In this section, you will go through these steps in sequence.

Creating a new ontology document

To create a new ontology document, click the New toolbar icon or select the command **File | New**.

Setting the ontology language level

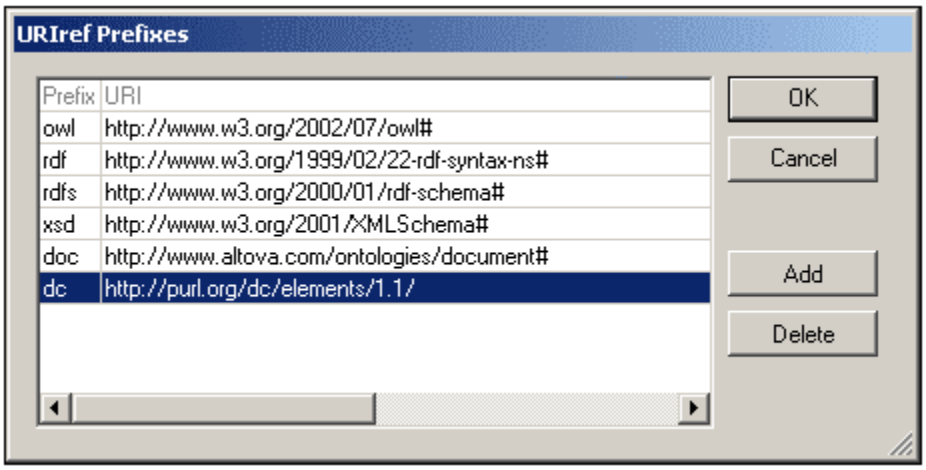
In the RDF/OWL Level combo box in the toolbar, select OWL DL. Alternatively, in the RDF/OWL menu, select RDF/OWL Level, and then OWL DL.

Declaring namespaces

For the OWL DL ontology you are creating you need to declare three namespaces:

- XML Schema namespace: `http://www.w3.org/2001/XMLSchema#`, to be declared with a prefix of `xsd`.
- A namespace for the Altova document vocabulary: `http://www.altova.com/ontologies/document#`, with a prefix of `doc`.
- The Dublin Core namespace for defining document metadata: `http://purl.org/dc/elements/1.1/`, with a prefix of `dc`.

Namespaces are declared in the URIref Prefixes dialog (menu item **Tools | URIref Prefixes**) shown below. Add a line for each required namespace using the Add button. Then enter the namespace and its prefix.



Note:

- It is important to declare namespaces at the very outset. This ensures that URIref prefixes used in the names of ontology items are correctly expanded to the relevant namespaces when a new ontology item is created. If a namespace has not been

 © 2006 Altova GmbH

declared, then the URIref prefix associated with that namespace, when used in the name of an item, will not be expanded to that namespace.
- The expansion of a URIref prefix to a namespace will not take place even if: (i) the relevant namespace is declared subsequent to the creation of an item; or (ii) if the ontology item is renamed after the namespace is declared but was created before the namespace was declared.
- Ontology items with unexpanded prefixes might not be correctly recognized.

Saving the file
You can save the file with a `.rdf` or `.owl` extension using the **File | Save** (**Ctrl+S**) command. Save the file with the name `AltovaDocuments.rdf`.

In the next section you will create the classes and define the class hierarchy.

4.2.2 Creating the Classes

In the document ontology, which makes use of the Altova Document vocabulary, we wish to create the following basic class hierarchy:

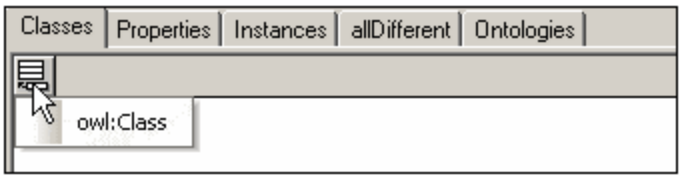
```
Document
|
|___PrintManual
|
|___WebPage
    |
    |___EnglishWebPage
    |
    |___GermanWebPage

Languages

OutputFormats
```

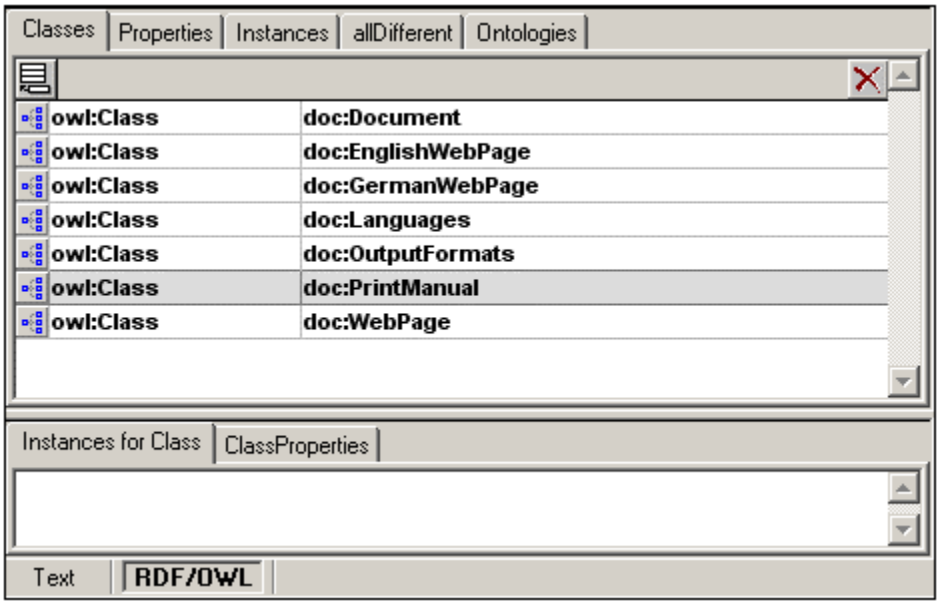
Creating the classes
To start with you will create the seven classes shown in the hierarchy above in the Classes Overview. The procedure for creating each class is as follows:

1. Click the Add New button in the Classes Overview to create an entry for the new class (*screenshot below*).



2. Type in the name of each class, using the `doc:` prefix for each. For example, `doc:Document`.

After you have finished creating these seven classes, the Classes Overview should look something like this:



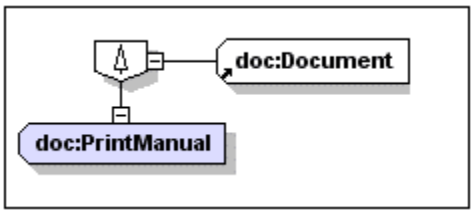
 © 2006 Altova GmbH

Check the validity of the document (correct syntax and partial consistency) by clicking the icon (**RDF/OWL | Semantics Check**). You should get messages saying the document is both well-formed and partially consistent.

Defining the class hierarchy for documents
Start by defining the class `PrintManual` as a subclass of the class `Document`. Do this as follows:

1. In the Classes Overview, click the Detail View button of the `PrintManual` class.
2. In the Detail View of `PrintManual`, right-click the `PrintManual` box, and select Add subClassOf.
3. Right-click the subClassOf connector box and select Add Class.
4. In the newly added class box, click the down arrow, and, from the dropdown list, select `doc:Document`.

When you are done, the Detail View of the `PrintManual` class will look like this:



Now create the following classes as subclasses using the method described above:

- `WebPage` as a subclass of `Document`.
- `EnglishWebPage` as a subclass of `WebPage`.
- `GermanWebPage` as a subclass of `WebPage`.

You have created a hierarchy for documents involving five of the seven classes you created. The other two classes (`Languages` and `OutputFormats`) are not directly involved in this hierarchy, and their definition will be described in the next section.

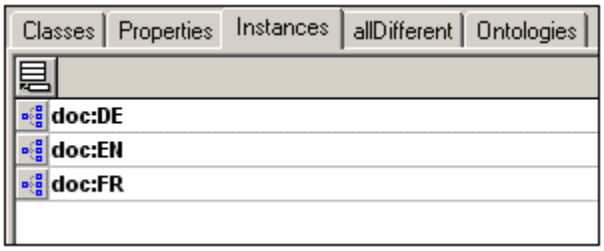
4.2.3 Instances as Class Enumerations

For the two classes, `Languages` and `OutputFormats`, we wish to enumerate their instances, specifically English (`EN`) and German (`DE`) for `Languages`, and HTML and PDF for `OutputFormats`. The way to do this is to first create instances for the classes (for example, `EN` and `DE` for `Languages`), and then specifying, for each class, that one of a list of enumerated instances is allowed as the instance of that class.

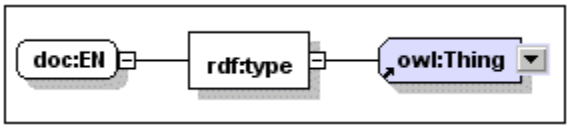
Creating instances

Start by creating instances for the `Languages` class. As follows:

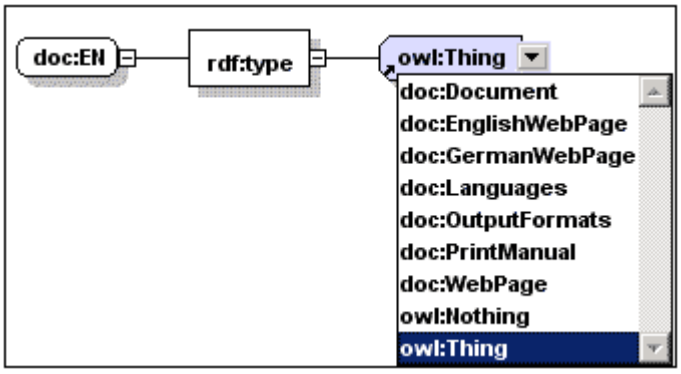
1. In the Instances Overview, click the Add New button to create an entry for the new instance. Name the instance `doc:EN`.
2. Repeat this step twice to create two more instances: `doc:DE` and `doc:FR`. The Instances Overview should now look like this:



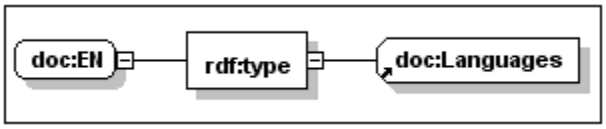
3. Click the Detail View button of the `EN` instance.
4. In the Detail View of the `EN` instance, expand the rdf:type predicate box, and double-click in the Resource Object box to pop up a down arrow on the right of the Resource Object box (*screenshot below*).



5. Click the down arrow to pop up a list containing available classes (*screenshot below*).



6. Select `doc:Languages`. This creates `EN` as an instance of the class `Languages`.



7. Define `DE` and `FR` as instances of the class `Languages` in exactly the same way you made `EN` an instance of `Languages`.

 © 2006 Altova GmbH

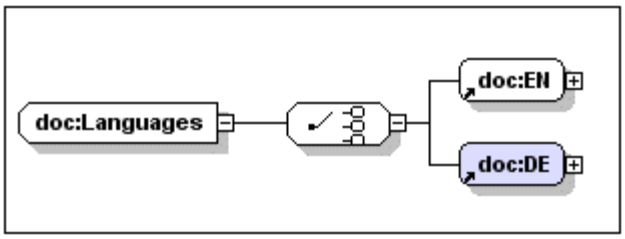
When you are done, you will have defined EN, DE, and FR as three instances of the class Languages.

Making instances the enumerations of a class

Now we wish to specify that the class Languages may have either the instance EN or DE as its value, but not FR. Do this as follows:

1. Click the Classes tab to go to Classes Overview, and there click the Detail View button of the Languages entry.
2. In the Detail View of Languages, right-click the Languages box, and select Add oneOf (which is not an OWL Lite feature but is an OWL DL feature).
3. Right-click the oneOf connector box and select Add Instance.
4. In the newly added instance box, click the down arrow, and, from the dropdown list, select doc:EN.
5. Right-click the oneOf connector box again, add another instance, and select doc:DE.

The Detail View of Languages should now look like this:



This indicates that the class Languages may be instantiated by one of the instances EN or DE.

Since doc:FR may not be an instance of of the Languages class, change its type to the general owl:Thing class (by double-clicking the Languages Resource Object box of doc:FR, and then clicking the down arrow and selecting owl:Thing from the dropdown list that appears).

Creating more enumerated classes

Now, using the same method as described above, create the instances HTML and PDF as enumerations of the class OutputFormats. Do this as follows:

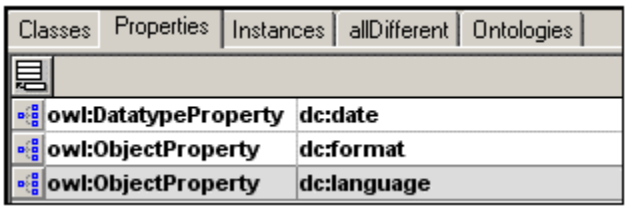
1. Create two new instances and name them doc:HTML and doc:PDF (in Instances Overview).
2. Define doc:HTML and doc:PDF as being instances of the OutputFormats class (in the Detail View of each instance separately).
3. Define the OutputFormats class to have HTML and PDF as its enumerated instances (in the Detail View of the OutputFormats class, and by using the oneOf connector).

Check the semantics of the ontology to ensure partial consistency.

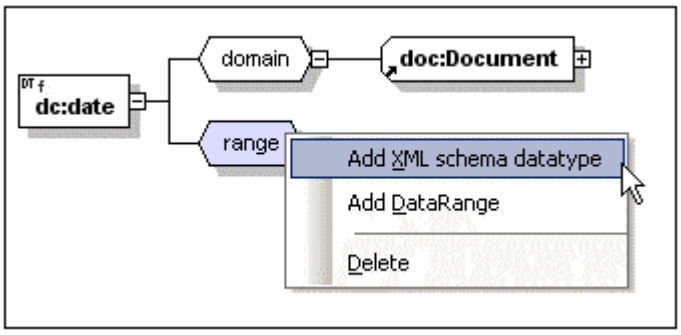
4.2.4 Defining the Properties

In this section you will define three properties to hold metadata information about documents. These properties are taken from the Dublin Core vocabulary for defining document metadata. They are the `dc:date`, `dc:language`, and `dc:format` properties. In this section, you will create the `dc:date` property as a datatype property with a range of the XML Schema datatype `date`, and the `dc:language` and `dc:format` datatypes as object datatypes with ranges, respectively of the `Languages` and `OutputFormats` classes. Create these three properties as follows:

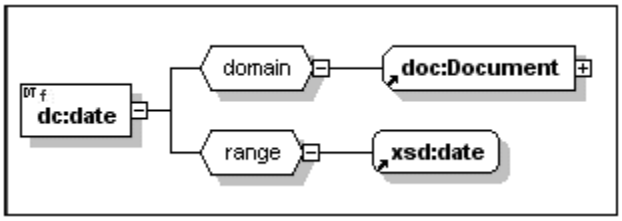
1. In the Properties Overview, click the Add New button to create an entry for a new datatype property. Name the property `dc:date`.
2. Repeat this step twice to create two more properties, but create both as object properties: `dc:language` and `dc:format`. The Properties Overview should now look like this:



3. Click the Detail View button of the `dc:date` property.
4. In the Detail View of the `dc:date` property, right-click the property box, select Domain, and set the Domain to the `doc:Document` class. Right-click the property box, select Range, then right-click the Range connector and select Add XML Schema Datatype (*screenshot below*). (The Add Data Range option creates an enumeration of data values (literals).)



Set the range to `xsd:date`. (If required, double-click in the Datatype box to get a list of available datatypes.) The Detail View should now look like this:



5. Set the domain and range for the `dc:language` and `dc:format` similarly. Set the domains of both to the `doc:Document` class, the range of `dc:language` to the `doc:Languages` class, and the range of `dc:format` to the `doc:OutputFormats` class.

Note: When the domain of a property is set to the `Document` class, all subclasses of the

 © 2006 Altova GmbH

`Document` class are also implicitly in the domain of that property. Consequently, the `PrintManual`, `WebPage`, `EnglishWebPage`, and `GermanWebPage` classes are also in the domain of properties that have their domain set to the `Document` class.

4.2.5 Describing Classes and Their Instances

In this section you will make class descriptions that specify restrictions and define detailed relationships between classes. Specifically, we wish to define that:

- The `WebPage` class is a union of the `EnglishWebPage` and `GermanWebPage` classes.
- The `WebPage` class has a restriction specifying that all its instances must have a `dc:format` property with an object that is the `HTML` instance.
- An instance of the `EnglishWebPage` class must be an instance of the `WebPage` class, with the restriction that its `dc:language` property have an object that is the `EN` instance.
- An instance of the `GermanWebPage` class must be an instance of the `WebPage` class, with the restriction that its `dc:language` property have an object that is the `DE` instance.

After creating these descriptions, you will create instances of these classes to test the validity of the axioms you have defined.

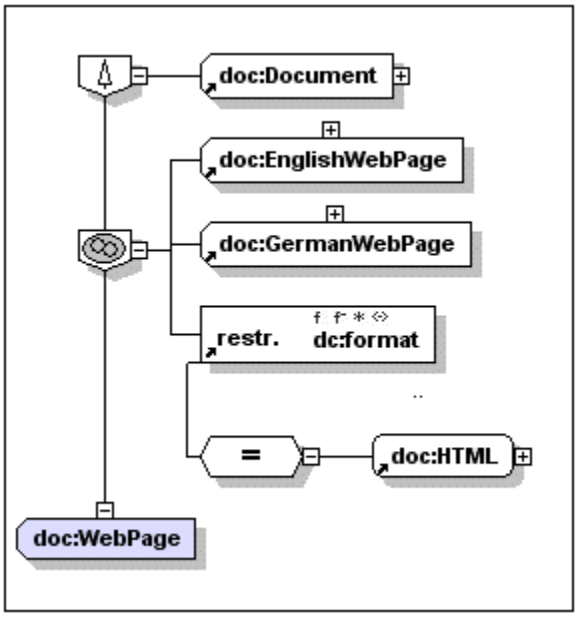
© 2006 Altova GmbH

Defining a class as a restricted union of classes

To define the `WebPage` class as a union of the `EnglishWebPage` and `GermanWebPage` classes, with a restriction that instances of the class have a `dc:format` property having a value that is the `HTML` instance, do the following:

1. In the Classes Overview, click the Detail View button of `WebPage`.
2. In the Detail View of `WebPage`, right-click the `WebPage` class box and select Add unionOf.
3. Right-click the unionOf connector, click Add Class from the context menu, and select `doc:EnglishWebPage`.
4. Right-click the unionOf connector, click Add Class from the context menu, and select `doc:GermanWebPage`.
5. Right-click the unionOf connector, click Add Restriction from the context menu, and select `dc:format`.
6. Right-click the Restriction box and select Add hasValue.
7. Right-click the hasValue box, click Add Resource Object, and select `doc:HTML`.

The Detail View should look like this:

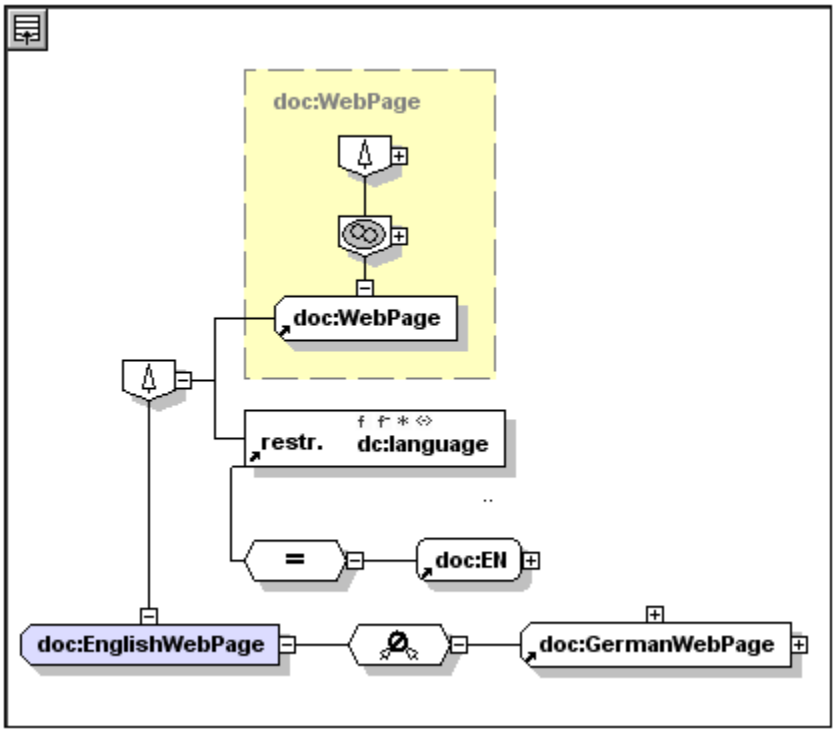


Defining a class as a restriction of another class

To define the `EnglishWebPage` class as an intersection of the `WebPage` class and the property `dc:language` having the `EN` instance as its object, do the following:

1. In the Classes Overview, click the Detail View button of `EnglishWebPage`.
2. In the Detail View of `EnglishWebPage`, right-click the subclassOf connector, click Add Restriction from the context menu, and select `dc:language`.
3. Right-click the Restriction box and select Add hasValue.
4. Right-click the hasValue box, click Add Resource Object, and select `doc:EN`.
5. Right-click the `EnglishWebPage` box, and select Add disjointWith.
6. Right-click the disjointWith box, click Add Class, and select `doc:GermanWebPage`.

The Detail View should look like this:



Define the `GermanWebPage` class similarly, with the difference that the `dc:language` restriction should be set to `doc:DE` and the class should be disjoint with the `EnglishWebPage` class.

 © 2006 Altova GmbH

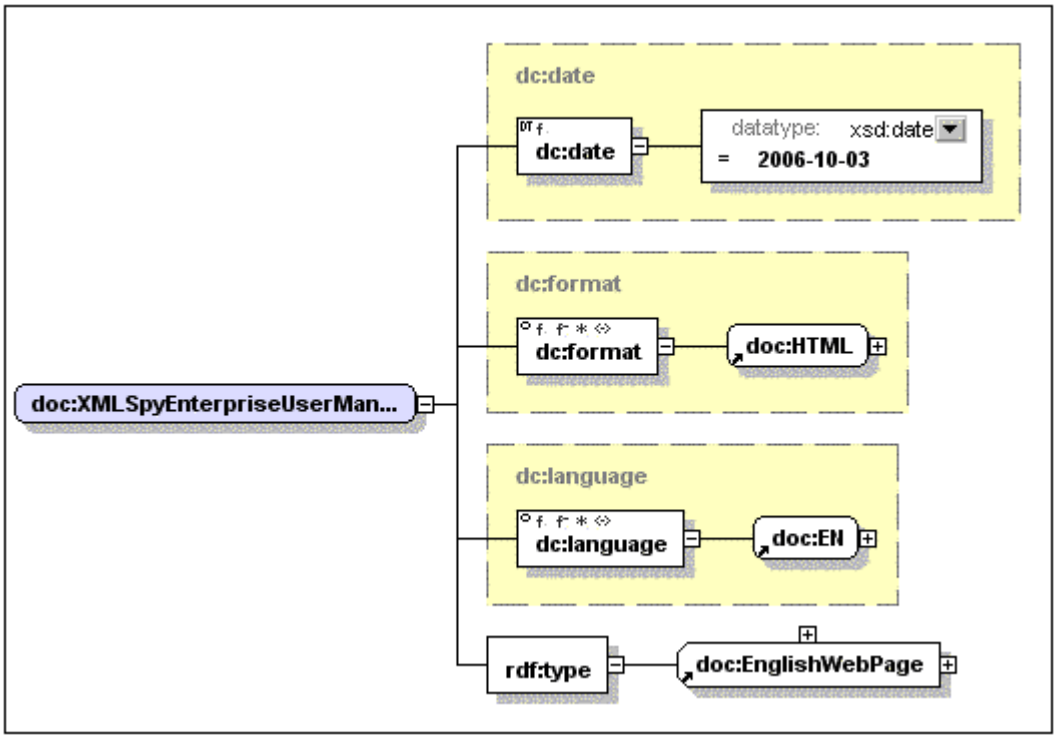
Creating instances of restricted classes
Let us create an instance to denote the index page of the HTML version of the English user manual of the XMLSpy Enterprise edition. Create and define this instance as follows:

1. In the Instances Overview, click the Add New button to create an entry for a new instance. Name the instance `doc:XMLSpyEnterpriseUserManualENHTML`.
2. In the Detail View of the `doc:XMLSpyEnterpriseUserManualENHTML` instance, expand the rdf:type predicate box, double-click in the Resource Object box to pop up a down arrow on the right of the Resource Object box, and select `doc:EnglishWebPage`.

Run a semantics check. You will get a message saying that the ontology appears to be inconsistent. This is because an instance of the `EnglishWebPage` class must have the `dc:format` and `dc:language` properties set, to `HTML` and `EN`, respectively. Make these settings, together with that for the `dc:date` property, as follows:

1. Right-click the Instance box, click Add Predicate from the context menu, and select `dc:format`.
2. Right-click the predicate box, click Add Resource Object from the context menu, and select `doc:HTML`.
3. Right-click the Instance box, click Add Predicate from the context menu, and select `dc:language`.
4. Right-click the predicate box, click Add Resource Object from the context menu, and select `doc:EN`.
5. Run a semantics check to confirm adequate consistency.
6. Right-click the Instance box, click Add Predicate from the context menu, and select `dc:date`.
7. Right-click the predicate box, click Add Literal Object from the context menu, and type `2006-10-03` in the newly created Literal Object box. Then click the datatype down arrow to display a dropdown list of available XML Schema datatypes, and select `xsd:date`.
8. Run a semantics check to confirm adequate consistency.

The Detail View should look like this:



Create and define an instance called `doc:XMLSpyEnterpriseUserManualDEHTML` similarly, with the difference that (i) this will be an instance of the `GermanWebPage` class, and (ii) the `dc:language` predicate should take `doc:DE` as its object.

© 2006 Altova GmbH

4.3 RDF Documents

In SemanticWorks, you can create RDF statements using the resources of ontologies referenced through the SemanticWorks interface. In the graphical RDF/OWL View, you create a new resource and then add predicate–object pairs for that resource. Predicates and objects can be selected in the SemanticWorks GUI from a list of resources made available via the referenced ontologies. The ontology resources are available in the SemanticWorks GUI as entry helpers, and are entered in the RDF document as URIrefs based on namespaces you declare for the RDF document.

In this part of the tutorial, you will create two RDF documents:

- Instances for an OWL DL Ontology, which provides a detailed description of the mechanism for creating, in a separate document, RDF resources based on resources from an ontology. This enables you to use an ontology on the Internet or a local network, as the basis of a separate RDF document.
- Creating a Dublin Core (DC) Document, which shows how you can create Dublin Core metadata for a resource such as a Web page or a book.

Note: If the resources of an ontology are not available, you can always directly type in URIrefs as required (in either Text View or RDF/OWL View).

4.3.1 Instances for an OWL DL Ontology

This RDF document is based on the OWL DL ontology (`AltovaDocuments.rdf`) you created in the previous part of this tutorial. The objective is to create instances of this OWL DL ontology in a separate RDF file. In order to be able to create such instances using ontology properties as predicates and ontology instances as objects, ontology resources must be availabe to the user via the GUI.

This part of the tutorial describes (i) how to set up the RDF document in SemanticWorks so that ontology resources are available via the GUI; and (ii) how to actually make RDF statements using these resources. This part of the tutorial is organized into three sections:

- Creating a New RDF Document;
- Referencing the Ontology;
- Making RDF statements

Note: You will need to know the location of the file `AltovaDocuments.rdf`, which you created in the previous section, OWL DL Ontology. If you did not do that part of the tutorial, you will find the file `AltovaDocuments.rdf` in the `Examples` folder of your SemanticWorks application folder.

Creating a New RDF Document

Create a new RDF document as follows:

1. Click the New toolbar icon or select the command **File | New**.
2. In the RDF/OWL Level combo box in the toolbar, select RDF. Alternatively, in the RDF/OWL menu, select RDF/OWL Level, and then RDF.
3. Save the document as `AltovaDocumentInstances.rdf`.

In the next section you will set up your document to reference the `AltovaDocuments.rdf` ontology and to use resources from this ontology in your RDF document.

Referencing the Ontology

The `AltovaDocuments.rdf` ontology needs to be referenced by the RDF document so that its resources (classes, properties, and instances) become available for use in the RDF document. The ontology referencing mechanism in SemanticWorks is implemented via a two-step procedure:

- Import namespaces from the ontology. In this step, each ontology namespace to be imported is listed, together with the location of the file. This is done in the Namespace Imports dialog (**Tools | Namespace Imports for RDF**).
- Declare the namespaces that will be used in the RDF document. All namespaces that will be used in the RDF document, including the imported namespaces, are declared, and prefixes are assigned for namespaces. This enables you to use the prefixes as shorthand for the namespace part of URIrefs. Namespaces are declared in the URIref Prefixes dialog (**Tools | URIref Prefixes**).

For your RDF document, you should carry out these two steps as described below.

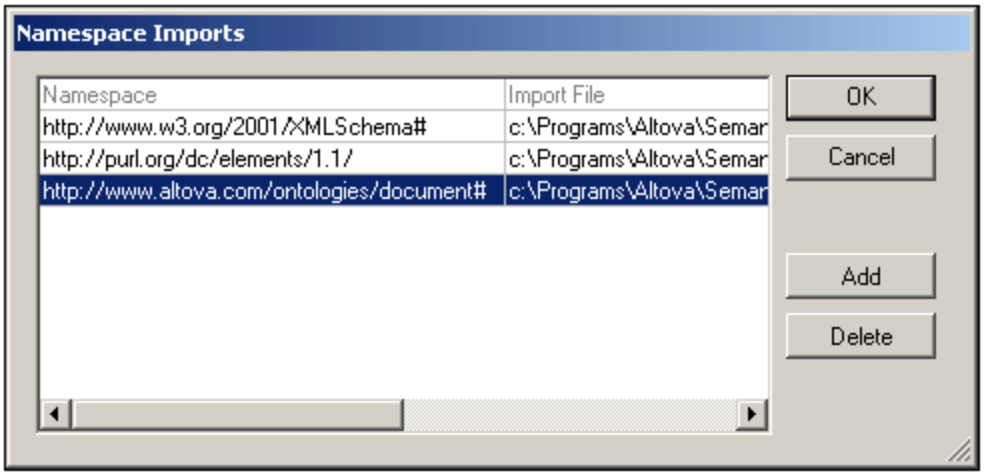
Importing namespaces from the ontology
The ontology you will be referencing is the OWL DL ontology `AltovaDocuments.rdf`, which you created in the previous part of this tutorial. This ontology uses three namespaces that you need to import into the RDF document:

 © 2006 Altova GmbH

- `http://www.w3.org/XMLSchema#`
- `http://www.altova.com/ontologies/document#`
- `http://purl.org/dc/elements/1.1/`

To import these namespaces, do the following:

1. Click **Tools | Namespace Imports for RDF**. This pops up the Namespace Imports dialog (*screenshot below*).

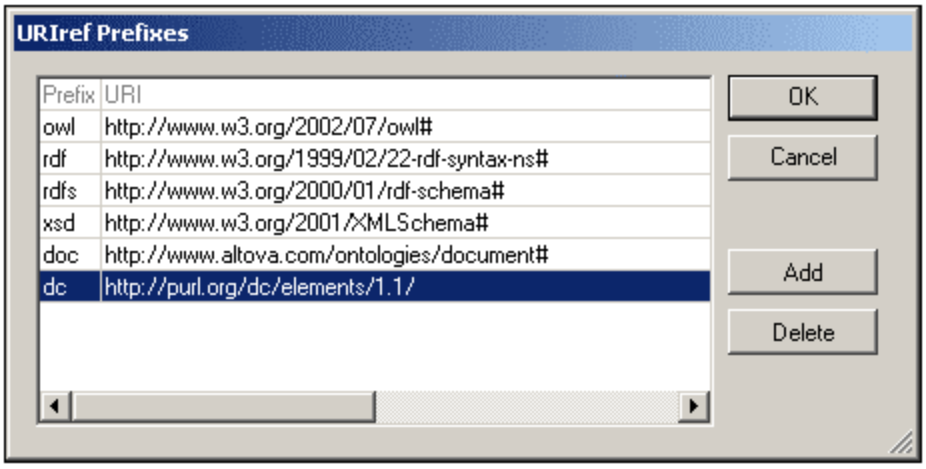


2. Add a line for a new entry by clicking the Add button, then enter the first namespace in the Namespace column and the location of the `AltovaDocuments.rdf` ontology in the Import File column. Note that you should give the absolute path for the ontology document.
3. Repeat Step 2 twice to add the next two namespaces and the ontology location. (The ontology location is the same for all three namespaces.)
4. When you are done, click OK.

Declaring namespaces for the RDF document
The RDF document you are creating uses three namespaces in addition to the RDF namespace:

- XML Schema namespace: `http://www.w3.org/2001/XMLSchema#`, to be declared with a prefix of `xsd`.
- A namespace for the Altova document vocabulary: `http://www.altova.com/ontologies/document#`, with a prefix of `doc`.
- The Dublin Core namespace for defining document metadata: `http://purl.org/dc/elements/1.1/`, with a prefix of `dc`.

Namespaces are declared in the URIref Prefixes dialog (**Tools | URIref Prefixes**) shown below. Add a line for each required namespace using the Add button. Then enter the namespace and its prefix.



You are now ready to start making RDF statements using resources from the `AltovaDocuments.rdf` ontology.

Troubleshooting
If the resources from the ontology do not appear in the Resources Overview, check the following:

- That RDF level is selected.
- That the namespaces are correctly entered. You should also check that they tally with the namespaces in the ontology document.
- That the location of the ontology file is an absolute path and is correctly entered.
- That namespaces have been correctly declared for the RDF document.

Making the RDF statements

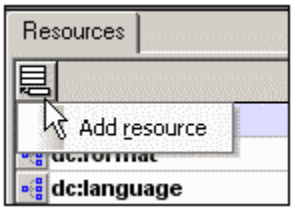
The mechanism for creating RDF statements in SemanticWorks consists of:

- Creating and naming the RDF resource (the subject) in the Overview of RDF/OWL View.
- In the Detail View of the resource, defining the predicates and objects of the resource.

You will now create resources for the various formats of the user manual of XMLSpy Professional Edition.

Creating and naming the RDF resource
To create a new RDF resource, do the following:

1. In the Overview of RDF/OWL View, click the Add New button, and select Add Resource (*screenshot below*).



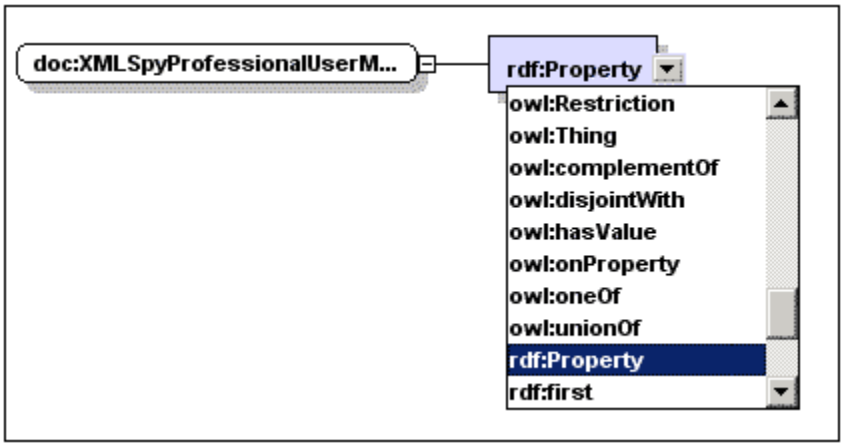
   An entry for the new resource is added to the list of resources.
2. Name the resource `doc:XMLSpyProfessionalUserManualENHTML` and press

 © 2006 Altova GmbH

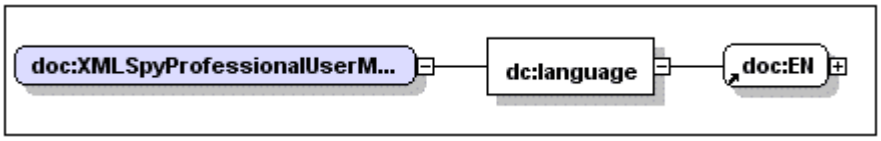
Enter. The resource is placed in the list according to its alphabetical order.

Defining the predicates and objects of the resource
The predicate and object of the newly created resource must be defined in the Detail View of the resource. Do this as follows:

1. Click the Detail View button of `doc:XMLSpyProfessionalUserManualENHTML` to go to its Detail View.
2. In Detail View, right-click the resource box and select Add Predicate.
3. Click the down arrow of the predicate box to display a list of all available resources (*screenshot below*). Select `dc:language`.



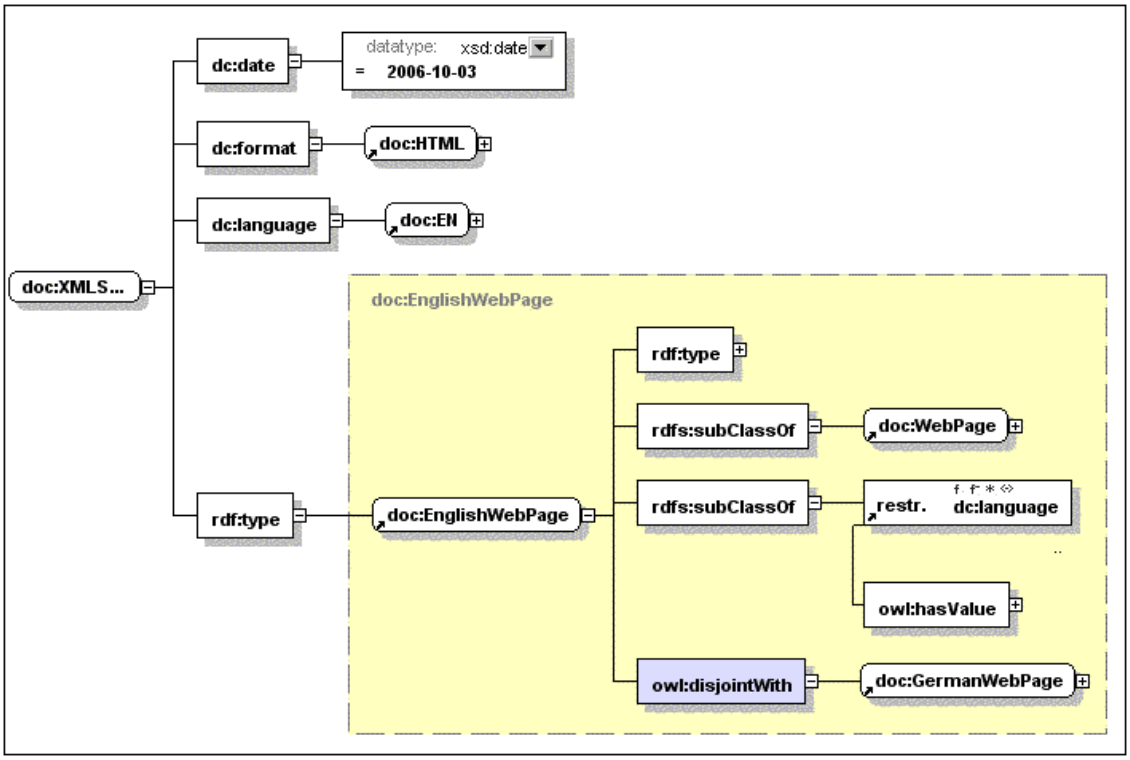
4. Right-click the `dc:language` box, and select Add Resource Object.
5. From the dropdown list of the Resource Object box, select `doc:EN`. (If the down arrow is not displayed at the right hand-side of the Resource Object box, double-click inside the box to display it.) The representation of the RDF statement should look like this:



You have now defined one property of your resource and its value.

6. To define the `dc:format` property, right-click the resource box, click Add Predicate, and select `dc:format` as the name of the predicate. Add a resource object to this property, and select `doc:HTML` (from the dropdown list) to be the resource object.
7. Create a `dc:date` property for the resource. Add a literal object—not a resource object —to the `dc:date` property, enter `2006-10-03` and select `xsd:date` as the datatype of the literal object.
8. Add another predicate to the resource and select `rdf:type` as its name. Add a resource object to the predicate and enter `doc:EnglishWebPage` as its name. When you expand the `doc:EnglishWebPage` box, you see its class relationships in detail (*screenshot below*); this is information from the referenced ontology.

The Detail View of the `XMLSpyProfessionalUserManualENHTML` resource will finally look something like this:



To complete the RDF document, create the following resources:

- `XMLSpyProfessionalUserManualDEHTML` (`language=DE`, `format=HTML`, `rdf:type=GermanWebPage`).
- `XMLSpyProfessionalUserManualENPDF` (`language=EN`, `format=PDF`, `rdf:type=PrintManual`).
- `XMLSpyProfessionalUserManualDEPDF` (`language=DE`, `format=PDF`, `rdf:type=PrintManual`).

Save the file and check the Text View of the document. Notice that only the newly created resources are defined in Text View. The list of resources in the Overview of RDF/OWL View, however, also includes the resources from the referenced ontology. This helps you to enter ontology resources quickly in RDF statements. Further, clicking the Detail View of an ontology resource causes the relationships of the resource to be displayed.

That's it!
You have learned how to quickly create RDF documents using the graphical interface of SemanticWorks.

© 2006 Altova GmbH

4.3.2 Creating a Dublin Core (DC) Document

To create a Dublin Core (DC) document in SemanticWorks, you will need an ontology of the DC vocabulary. The `Examples` folder of your SemanticWorks application folder contains an OWL Lite ontology of the DC vocabulary, called `DCOntology.rdf`.

Creating a DC document involves the following two steps, both of which are described in detail in the respective subsections:

- Referencing the DC Ontology;
- Creating the DC Metadata.

Note on the Dublin Core ontology delivered with SemanticWorks
The following points should be noted:

- The ontology is an OWL Lite ontology
- The DC vocabulary covered is the DC Simple set of 15 basic elements.
- The DC elements have been created as properties in the ontology.
- No datatypes have been defined for DC elements. If you wish to assign datatypes, then select the required datatype in the GUI when entering the object definition. (Note that defining the datatype in the ontology is not sufficient to create the metadata in the RDF document as that datatype.)

Note on the Dublin Core template delivered with SemanticWorks
The following points should be noted:

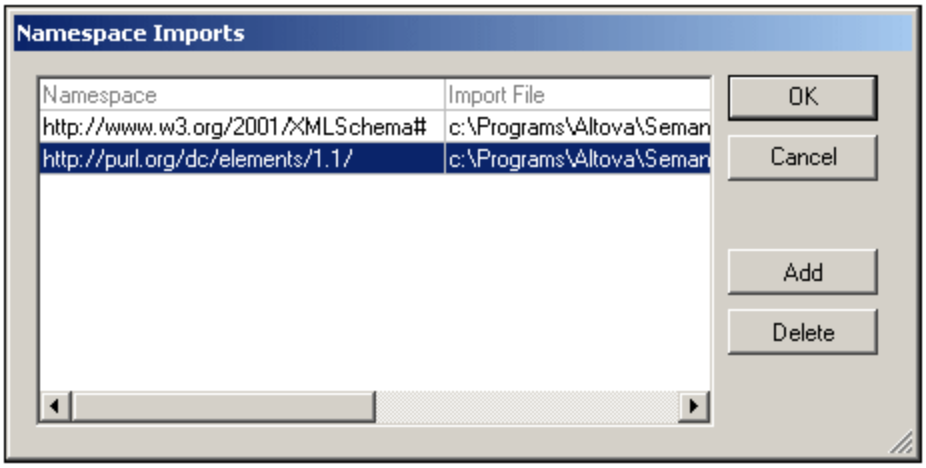
- The DC template is called `DCTemplate.rdf` and is located in the `Examples` folder of your SemanticWorks application folder.
- The template is intended to be used as a starting point for building RDF documents providing metadata while using the DC vocabulary.
- The template references the DC ontology (`DCOntology.rdf`) using an absolute path. You will need to change this absolute path so that it correctly points to the file `DCOntology.rdf` in the `Examples` folder of your SemanticWorks application folder.

Referencing the DC Ontology

After opening a new document, select the RDF level (**RDF/Owl | RDF/OWL Level**) and save the document as `DCMetadataSample.rdf`.

Referencing the DC ontology
To reference the Dublin Core (DC) ontology, do the following:

1. Check that you are in RDF level.
2. Click **Tools | Namespace Imports for RDF**. This pops up the Namespace Imports dialog (*screenshot below*).

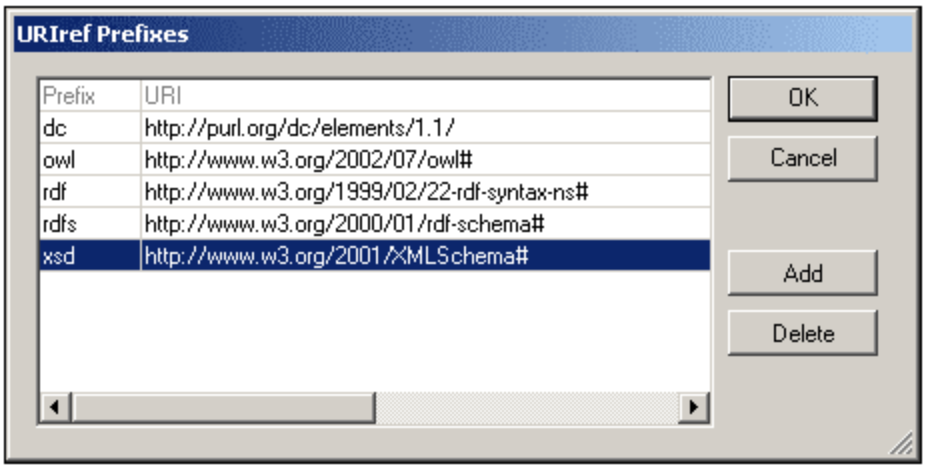


3. In the Namespace column enter `http://purl.org/dc/elements/1.1/`, which is the DC namespace declared in the ontology file `DCOntology.rdf`.
4. In the Import File column, enter the location of the ontology file `DCOntology.rdf`. This file will have been delivered in the `Examples` folder of the SemanticWorks application folder. Note that you should give the absolute path for the ontology document.
5. If you have edited the ontology file to define XML Schema datatypes, you will need to import the XML Schema namespace (`http://www.w3.org/2001/XMLSchema#`), too. Enter the namespace to be imported in the Namespace column and the location of the ontology file `DCOntology.rdf` in the Import File column.
6. Click OK to complete.

Note: Defining a DC element in the ontology as having a certain datatype does not automatically insert that datatype annotation when that DC element is inserted in the RDF document. The datatype information in the RDF document must be explicitly entered when entering the object definition for that metadata.

Declaring namespaces for the RDF document

Your RDF document will require the DC (`http://purl.org/dc/elements/1.1/`) and XML Schema (`http://www.w3.org/2001/XMLSchema#`) namespaces to be declared. Declare these namespaces, with prefixes, in the URIref Prefixes dialog (**Tools | URIref Prefixes**) shown below. Add a line for each namespace using the Add button. Then enter the namespace and its prefix. Click OK to complete.

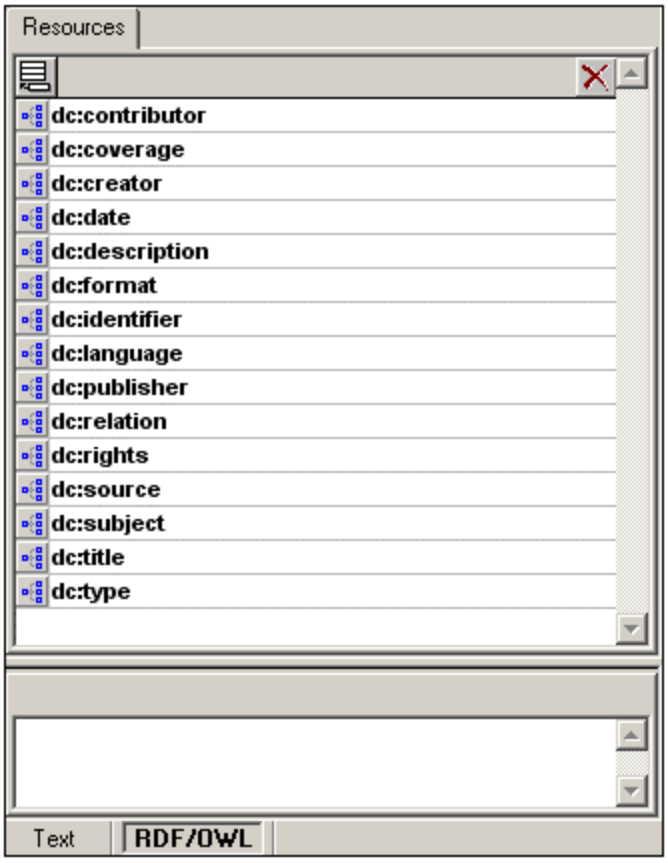


 © 2006 Altova GmbH

Note:
The following points about namespaces in the RDF document should be noted.

- If you do not wish to use datatyping in an RDF document, there is no need to declare the XML Schema namespace. If, in this case, the ontology contains datatype definitions, then these will appear in the GUI as expanded URIrefs. To collapse the namespace part of the URIref, you should then declare the XML Schema namespace with a prefix.
- If you wish to create resources in a specific namespace, you must declare this namespace.

After you have imported the DC ontology namespaces and declared namespaces for the RDF document, the RDF/OWL View Overview should look like this:



The 15 Simple DC elements are now available as resources and can be used as predicates of RDF statements. You are now ready to start creating the DC metadata.

Troubleshooting
If the DC resources do not appear in the Resources Overview, check the following:

- That RDF level is selected.
- That the namespaces are correctly entered. You should also check that they tally with the namespaces in the ontology document.
- That the location of the ontology file is an absolute path and is correctly entered.
- That namespaces have been correctly declared for the RDF document.

Creating the DC Metadata

In this section, you will create Dublin Core (DC) metadata for a single resource. Specifically, you will create a resource called *A Sample Page*, and define `dc:title`, `dc:description`, and `dc:date` elements for it.

Creating a new resource

To create a new resource, create the Add New button (*screenshot below*), and name the newly created resource `urn:SamplePage`.

Adding DC metadata for a resource

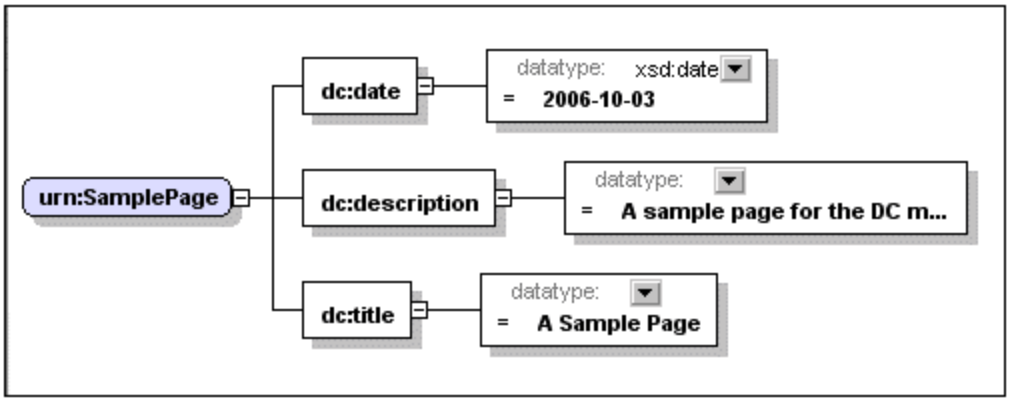
Click the Detail View button of `urn:SamplePage` to go to the Detail View of `urn:SamplePage` (*screenshot below*).

Now do the following:

1. Right-click the `urn:SamplePage` box, click Add Predicate, and from the dropdown list select `dc:title`. Right-click the `dc:title` box, click Add Literal Object, and type `A Sample Page` in the newly created Literal Object box. The `dc:title` metadata is created.
2. Right-click the `urn:SamplePage` box, click Add Predicate, and from the dropdown list select `dc:description`. Right-click the `dc:description` box, click Add Literal Object, and type `A Sample Page for the DC metadata tutorial` in the newly created Literal Object box. The `dc:description` metadata is created.
3. Right-click the `urn:SamplePage` box, click Add Predicate, and from the dropdown list select `dc:date`. Right-click the `dc:date` box, click Add Literal Object, and type `2006-10-03` in the newly created Literal Object box. Then click the datatype down arrow to display a dropdown list of available XML Schema datatypes, and select `xsd:date`. The `dc:date` metadata is created and has the XML Schema datatype `xsd:date`.

The Detail View of `urn:SamplePage` should look something like this:

 © 2006 Altova GmbH



Switch to Text View to see the RDF/XML serialization, which should look something like this:

```
<?xml version="1.0"?>
<rdf:RDF xml:base="file:///C:/workarea/SemanticWorks/tutorial/DCTemplate.rdf"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:owl="http://www.w3.org/2002/07/owl#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
    <rdf:Description rdf:about="urn:SamplePage">
        <dc:date rdf:datatype="xsd:date">2006-10-03</dc:date>
        <dc:description>A sample page for the DC metadata tutorial</dc:description>
        <dc:title>A Sample Page</dc:title>
    </rdf:Description>
</rdf:RDF>
```

Notice that the `dc:date` element has an `rdf:datatype` attribute with a value of `xsd:date`.

You can create as many DC elements as predicates of this resource that you want. To create more resources, go back to RDF/OWL View Overview. Save the file to complete this part of the tutorial.

That's it!
You have learned how to create a Dublin Core RDF document using the graphical interface of SemanticWorks. In the `Examples` folder of the SemanticWorks application folder, you will find a DC ontology (`DCOntology.rdf`) and a DC template (`DCTemplate.rdf`) to help you create DC RDF documents.

Chapter 5

User Reference

5 User Reference

This **User Reference** section describes all the icons that appear in the toolbars and Detail View and the commands in SemanticWorks menus. It ends with a section on usage issues. The User Reference is organized into the following subsections:

- Toolbar Icons
- Icons in Detail View
- File menu commands
- Edit menu commands
- View menu commands
- RDF/OWL menu commands
- Tools menu commands
- Window menu commands
- Help menu commands
- Usage Issues

© 2006 Altova GmbH

5.1 Toolbar Icons

Icons in the toolbar are shortcuts for various commands, all of which are also available as menu commands. In this section, the icons are listed together with brief descriptions of the commands for which they are shortcuts. The commands are described in more detail in the corresponding menu section in the User Reference.

The toolbar icons are arranged in the following groups:

- Main
- View Options
- Classes
- Properties
- Miscellaneous

Main
The Main group of icons are shortcuts to basic file and editing commands.

New (**File** menu, **Ctrl+N**)
Creates a new RDF document with a name of `UntitledX`, where `X` is an integer.

Open (**File** menu, **Ctrl+O**)
Pops up the Open dialog, in which you can browse for the file to be opened.

Save (**File** menu, **Ctrl+S**)
Saves the active document to file. Enabled only if the active document has been modified.

Print (**File** menu)
Displays the Print dialog (for printing the Detail View of the selected item).

Undo (**Edit** menu, **Ctrl+Z**, **Alt+Backspace**)
Undoes an editing change.

Redo (**Edit** menu, **Ctrl+Y**)
Redoes an undo.

Cut (**Edit** menu, **Ctrl+X**, **Shift+Delete**)
Cuts the selected text (in Text View) from the document and saves it to the clipboard.

Copy (**Edit** menu, **Ctrl+C**)

Copies the selected text (in Text View) to the clipboard.

Paste (**Edit** menu, **Ctrl+V**)
Pastes the clipboard text into the active document at the cursor position.

Find (**Edit** menu, **Ctrl+F**)
Finds the submitted text string (in Text View).

Find Next (**Edit** menu)
Finds the next occurrence of the submitted text string (in Text View).

View Options
The View Options group of icons are shortcuts to commands that affect the view and the semantics of the active document in the GUI.

Show/Hide Blank Nodes (**View** menu)
A toggle command that shows/hides blank nodes in the Overview categories.

Show Possible Inconsistencies (**View** menu)
A toggle command to show/hide possible semantic inconsistencies in the ontology. Applies to the semantics check on OWL Lite and OWL DL ontologies.

RDF/OWL Level (**RDF/OWL** menu)
Selects the RDF/OWL level for the active document: RDF, RDF Schema, OWL Lite, OWL DL, or OWL Full. The selected level will apply till changed or till the document is closed.

Syntax Check (**RDF/OWL** menu)
Checks the syntax of the active document.

Semantics check (**RDF/OWL** menu)
Checks the semantics of the active OWL Lite or OWL DL ontology document.

Reload All Imports (**RDF/OWL** menu)
Reloads all ontologies that the active document imports.

Expand URIref Prefixes (**Tools** menu)

 © 2006 Altova GmbH

A toggle command to expand/use URIref prefixes in the serialized RDF/XML.

Classes
The Classes group of icons enables you to add a relationship to a class. These commands are also available in the context menu that appears when the box of an eligible class is right-clicked in Detail View.

Adds subClassOf
Adds a `subClassOf` relationship to a class.

Adds intersectionOf
Adds an `intersectionOf` relationship to a class.

Adds unionOf
Adds a `unionOf` relationship to a class.

Adds complementOf
Adds a `complementOf` relationship to a class.

Adds oneOf
Adds a `oneOf` relationship to a class.

Adds disjointWith
Adds a `disjointWith` relationship to a class.

Add equivalentClass
Adds an `equivalentClass` relationship to a class.

Properties
The Properties group of icons enables you to define certain attributes of properties. These commands are also available in the context menu that appears when a property box is right-clicked in Detail View.

Adds subPropertyOf
Adds a `subPropertyOf` relationship to a property.

Adds domain

Adds a domain for a property.

Adds range
Adds a range specifier for a property.

Add equivalentProperty
Adds an `equivalentProperty` relationship to a property.

Add inverseOf
Adds an `inverseOf` relationship to a property.

Miscellaneous
The Classes group of icons enables you to add a relationship to a class. These commands are also available in the context menu that appears when the box of an eligible class is right-clicked in Detail View.

Adds resource object
Adds a resource object to a predicate.

Adds literal object
Adds a literal object to a predicate.

Adds predicate
Adds a predicate to an RDF resource.

Adds restriction
Adds a restriction to a class or property relationship.

Adds allValuesFrom
Adds the `allValuesFrom` relationship to link a restriction to a class or data range.

Adds someValuesFrom
Adds the `someValuesFrom` relationship to link a restriction to a class or data range.

Adds hasValue
Adds the `someValuesFrom` relationship to link a restriction to a class instance or a

 © 2006 Altova GmbH

data value.

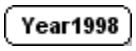
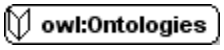
5.2 Icons in Detail View

The various icons used to display relationships between ontology items in Detail View are listed below, with a brief description. Icons are organized into the following groups:

- Ontology items
- RDF containers and collections
- Class descriptions
- Class axioms
- Property descriptions
- OWL individuals (instances)

Ontology items

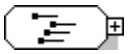
Ontology items are classes, instances, properties, and literals. In the case of some items, variants are distinguished.

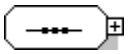
	The Class icon is used for both RDFS and OWL classes. Contrast the bevelled edges on the left of the Class icon with the rounded edges of the Instances icon.
	Instances of RDFS and OWL classes, and subjects of RDF Triples. (Instances are also known as Individuals in OWL terminology.)
	RDFS property. Distinguished from OWL properties by the lack of symbols in the top left-hand corner.
	OWL object property. Distinguished from OWL datatype properties by the O symbol at extreme top left. The other symbols, from left to right, are: functional property, inverse functional property, transitive property, and symmetric property. Clicking a symbol sets the property to that type.
	OWL datatype property. Distinguished from OWL object properties by the DT symbol at extreme top left. The property type can be set to functional by clicking the f symbol.
	OWL ontology. The ontology header is optional. It is useful for importing other ontologies and for declaring prior versions.

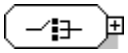
../contd.

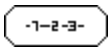
 © 2006 Altova GmbH

RDF containers and collections
The following icons indicate class relationships, and can be inserted when a class is selected.

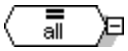

rdf:Bag.


rdf:Seq.

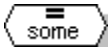

rdf:Alt.


rdf:List.

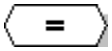
Class descriptions
The following icons indicate class relationships, and can be inserted when a class is selected.



owl:allValuesFrom. Specifies that all values allowed on a restriction must come from the specified class or data range.



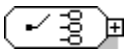
owl:someValuesFrom. Specifies that at least one value allowed on a restriction must come from the specified class or data range.


owl:hasValue. Specifies the value that a restriction must take.

owl:unionOf. A class that is a union of two or more classes is equal to that union.

owl:intersectionOf. When a class ABC containing A's, B's, and C's intersects with a class CDE containing C's, D's, and E's, the resulting class contains C's.

owl:complementOf. When a class A is a complement of class B, then no instance of A can be an instance of B.


owl:oneOf. Describes a class by enumerating its instances.

../contd.

Class axioms
The following icons indicate OWL class relationships, and can be inserted when a class is selected.

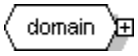
`rdfs:subClassOf`. The selected class is a subclass of another class.

`owl:equivalentClass`. Declares the equality of the selected class with another class.

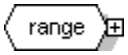
`owl:disjointWith`. Declares the inequality of the selected class with other classes.

Property descriptions
The following icons indicate OWL class relationships, and can be inserted when a class is selected.

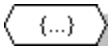
`rdfs:subPropertyOf`. Declares the selected property as a subproperty of another property. For example, the property `#hasMother` could be a subproperty of the property `#hasParent`.



`rdfs:domain`. Specifies the domain of a property `P`, i.e. the class of resources that may be the subject in a triple with the predicate `P`.



`rdfs:range`. Specifies the range of a property `P`, i.e. the class of resources (or datatypes) that may be the value, in a triple, of the predicate `P`.

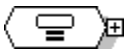


`owl:DataRange`. Defines an enumeration of data values.

`owl:equivalentProperty`. Declares equivalence between properties. Equivalent properties have the same property extensions.

`owl:inverseOf`. Declares one property to be the inverse of another. For example, the property `#hasChild` could be the inverse of the property `#hasParent`.

OWL individuals (instances)
The following icons indicate OWL class relationships, and can be inserted when a class is selected.


`owl:sameAs`. Declares two individuals to be identical.

`owl:differentFrom`. Declares the inequality of two individuals.

`owl:AllDifferent`. Declares the pairwise inequality of all individuals in a group.

© 2006 Altova GmbH

5.3 File Menu

The commands in the **File menu** enable you to create, open, save, export, and print SemanticWorks files. (SemanticWorks files are files with the `.nt`, `.rdf`, `.rdfs`, and `.owl` file extensions.)

New (Ctrl+N)
Opens a new blank document in the GUI with a name of `UntitledX` (where `X` is an integer). This document can subsequently be saved as an RDF (`.rdf`), RDF Schema (`.rdfs`), or OWL (`.owl`) file, or in the XML (`.xml`) or text (`.txt`) formats. To save to the N-Triples (`.nt`) format, use the **File | Export to `.nt`** command. Note that since all OWL files are valid RDF files, they are typically saved as `.rdf` files. The new document is created with the following rudimentary content:

```
<?xml version="1.0"?>
<rdf:RDF  xmlns:owl="http://www.w3.org/2002/07/owl#"
          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"/>
```

Notice that the RDF, RDF Schema, and OWL namespaces are automatically declared on the `rdf:RDF` element.

Open (Ctrl+O)
Opens `.rdf`, `.rdfs`, `.owl` , `.nt`, and `.xml` files in the RDF/OWL view.

Save (Ctrl+S), Save As
Saves the active document as an RDF (`.rdf`), RDF Schema (`.rdfs`), OWL (`.owl`) file, or in the XML (`.xml`) or text (`.txt`) formats. To save to the N-Triples (`.nt`) format, use the **File | Export to `.nt`** command. Note that once a file is saved in a particular format, it is available only in that format and can, therefore, only be viewed in that format in other editors. For example, a `.nt` file can be viewed in a standard text editor in N-Triples format only; the graphical view of this `.nt` file are features special to SemanticWorks.

Save Diagram as Image
This command is active in the Detail View of RDF/OWL View and saves the active Detail View document as an image in the PNG format or any desired image format.

Export to `.nt` and `.xml`
Exports the active SemanticWorks document as an N-Triples or XML file to the desired location. The file is exported with the `.nt` or `.xml` file extension, respectively.

Close, Close All
Closes, respectively, the active document and all open documents.

Encoding
Pops up the Encoding dialog, in which you can set the encoding of the RDF/XML, RDF Schema, or OWL document. The encoding you select is entered as the value of the encoding attribute of the XML declaration of the document, as in `<?xml version="1.0" encoding="UTF-8"?>`. Note that default encoding is set in the Encoding tab of the Options dialog (**Tools | Options**).

Print, Print Preview, Print Setup
The Print and Print Preview commands are available for Text View and Detail View, and enable these two views to be printed. The Print Setup command enables you to configure a printer for the print job.

Recently Used Files
Displays the four most recently opened documents.

Exit
Closes all open documents and exits the application. If a document has unsaved changes, you are prompted about whether you wish to save the changes.

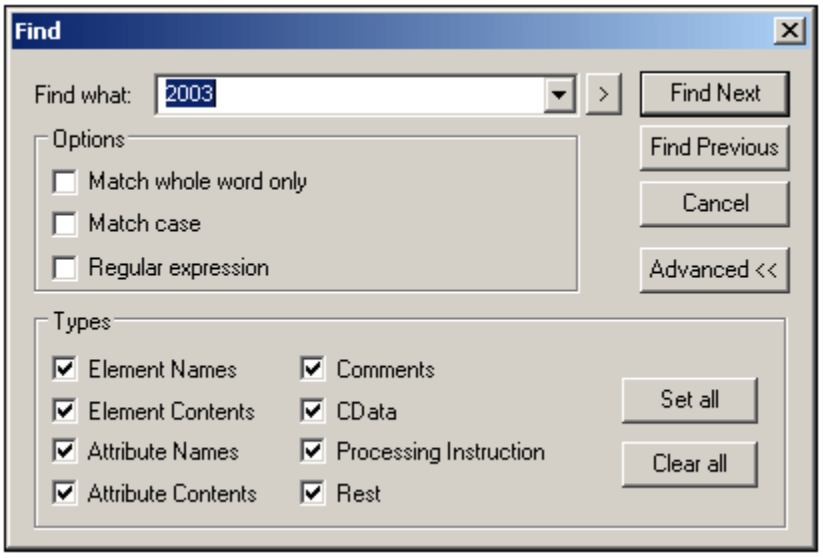
© 2006 Altova GmbH

5.4 Edit Menu

The commands in the **Edit menu** enable you to navigate and edit documents in the SemanticWorks interface quickly and efficiently.

Undo (Ctrl+Z), Redo (Ctrl+Y)
Respectively, undoes and redoes the previous command. Both commands can be used a multiple number of times in sequence in order to undo or redo multiple steps. The command is available in both Text View and RDF/OWL View.

Find (Ctrl+F), Find Next
In Text View or Detail View, finds the input text string if that string is present in the current view. You can select options to match the input string as a whole word and/or whether the search should be case-sensitive. In Text View (*for which, screenshot of the Find dialog is shown below*), you can additionally search using regular expressions. Also, in Text View, when you click the Advanced button, you can select what parts of the XML document are searched.



The Find Next command finds the next instance of the search string.

Replace
In Text View, pops up the Find and Replace dialog in which you can specify a text string `A` to find and a text string `B` with which to replace the text string `A`. The options for specifying the text string to find are as described for the Find command.

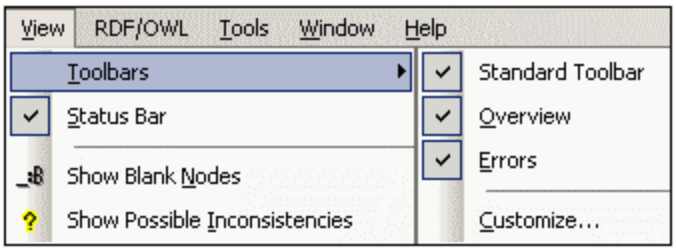
Cut (Shift+Delete), Copy (Ctrl+C), Paste (Ctrl+V)
In Text View, respectively, cuts or copies the selected text to the clipboard, and pastes from the clipboard to the cursor position in Text View.

Delete (Delete)
In RDF/OWL View, deletes the selected object.

5.5 View Menu

The commands in the **View menu** enable you to configure the display of toolbars and the Status Bar, and enable you to toggle on or off the display of blank nodes (anonymous classes).

Toolbars
The Toolbars menu item pops out a submenu (*screenshot below*) in which you can choose, individually, whether the Standard Toolbar, Overview Window, and Errors Window should be displayed or not. When one of these submenu items is checked, it is displayed in the GUI; otherwise, it is hidden.



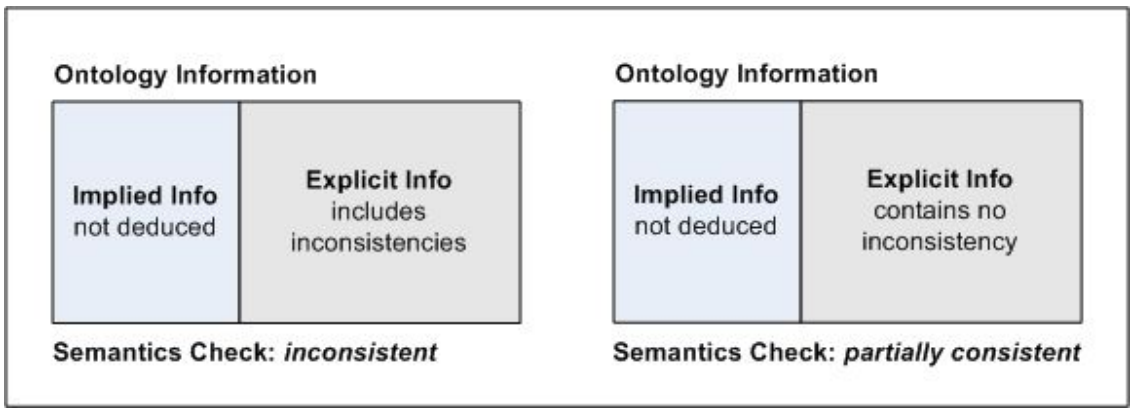
The Customize submenu item pops up the Customize dialog in which you can customize menu commands, toolbar options, and shortcut options.

Status Bar
The Status Bar menu item toggles on and off the display of the Status Bar located at the bottom of the SemanticWorks application window. The Status Bar displays a short description of menu items and toolbar icons when the mouse is placed over such an item.

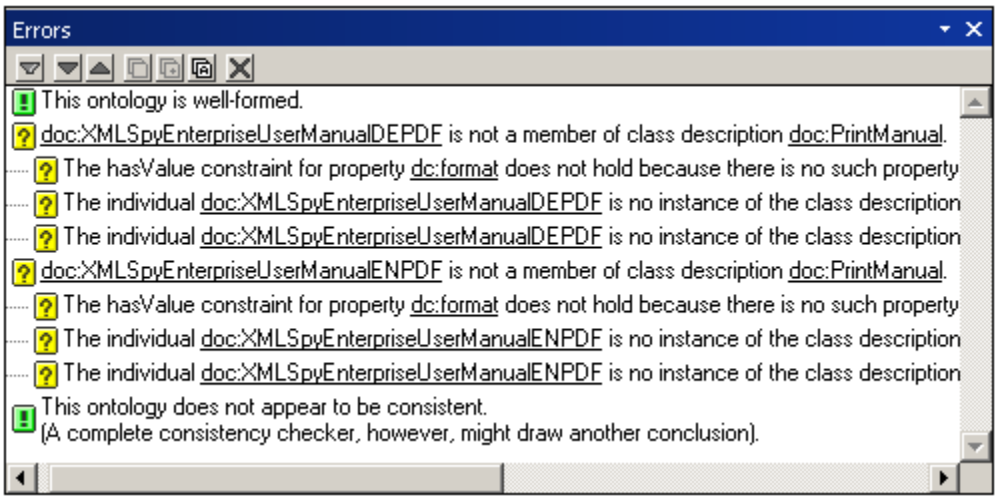
Show Blank Nodes
Toggles the display of blank nodes (anonymous classes) on and off. When selected, the toggle is on (blank nodes are displayed); when unselected, the toggle is off (blank nodes are not displayed).

Show Possible Inconsistencies
A toggle command to show possible semantic inconsistencies within OWL Lite and OWL DL ontologies. The semantic check in SemanticWorks is a partial semantic check. It is based on knowledge explicitly stated in the ontology; implied knowledge is not deduced. This means that implied knowledge, such as that derived through entailment or inference, will not be evaluated when the ontology is checked for its semantics. The semantics check, therefore, checks for inconsistencies in the explicit knowledge. The relationship between inconsistencies and the semantics check of SemanticWorks is shown in the illustration below. (*Also see the section Semantics Check* in the User Reference.)

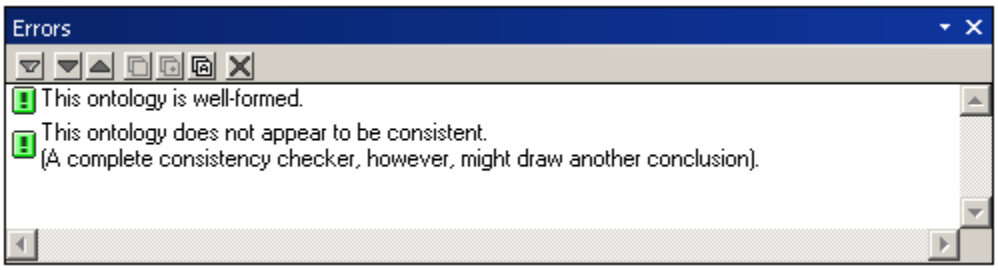
© 2006 Altova GmbH



When the Show Possible Inconsistencies toggle is switched on, inconsistencies that arise because implied knowledge is not used are displayed in the Errors Window (*screenshot below*).


Errors
This ontology is well-formed.
doc:XMLSpyEnterpriseUserManualDEPDF is not a member of class description doc:PrintManual.
The hasValue constraint for property dc:format does not hold because there is no such property
The individual doc:XMLSpyEnterpriseUserManualDEPDF is no instance of the class description
The individual doc:XMLSpyEnterpriseUserManualDEPDF is no instance of the class description
doc:XMLSpyEnterpriseUserManualENPDF is not a member of class description doc:PrintManual.
The hasValue constraint for property dc:format does not hold because there is no such property
The individual doc:XMLSpyEnterpriseUserManualENPDF is no instance of the class description
The individual doc:XMLSpyEnterpriseUserManualENPDF is no instance of the class description
This ontology does not appear to be consistent.
(A complete consistency checker, however, might draw another conclusion).

When the toggle is switched off, possible inconsistencies are not displayed (*screenshot below*).


Errors
This ontology is well-formed.
This ontology does not appear to be consistent.
(A complete consistency checker, however, might draw another conclusion).

Note: The display of inconsistencies can also be switched on and off via the Inconsistency Warnings filter in the Filter Menu of the Errors Window.

5.6 RDF/OWL Menu

The **RDF/OWL menu** contains commands that enable you to make settings for RDF/OWL document editing and checking and commands to carry out these checks.

RDF/OWL Level
Pops out a submenu from which you can select the RDF or ontology specification (RDF Schema, OWL Lite, OWL DL, or OWL Full) according to which the active document should be edited. The RDF/OWL Level can also be selected in the corresponding combo box in the View Options toolbar (*screenshot below*).

This step is important because it sets up the GUI for appropriate editing interaction. For example, when RDF is selected, the Main Window contains only a Resources Overview as opposed to the Overview of five categories when one of the ontology levels is selected. Further, the insertion of items appropriate for the selected RDF/OWL level depends on the selection you make.

When an existing document is opened, or when a new document is created, the OWL Full level is selected by default. After you change the level to the required level, this level is maintained as long as the document is open or till the level is changed. You can change levels as many times as you wish. The display of the document will change accordingly.

Note: When the OWL Lite or OWL DL level is selected, you can additionally run syntax and semantics checks on the document for that level (i.e. against the specification corresponding to the selected level).

Reload All Imports
Reloads all imported ontologies in the active document. This command is useful if you have modified an imported ontology after opening the active document.

Syntax Check
The Syntax Check command checks the syntax of the active document according to syntax rules specified in the corresponding specification. The result of the check (positive = well-formed) is displayed in the Errors Window pane. Additionally, if errors are detected, these are listed in the Errors Window pane and include links to the Detail View of the offending items.

Semantics Check
The Semantics Check command is enabled when the active document has an RDF/OWL level that is OWL Lite or OWL DL. It checks the semantics of the active document according to semantics rules specified in the corresponding specification. The result of the check (positive = at least partially consistent) is displayed in the Errors Window pane. Additionally, if errors are detected, these are listed in the Errors Window pane and include links to the Detail View of the items that lead to the inconsistencies.

Note that the semantics check is a partial semantics check, which means that only knowledge stated explicitly in the ontology is evaluated, while knowledge implicit in certain ways (such as from entailment) is not evaluated. Inconsistencies arising from the non-consideration of possibly existent implicit information are displayed when the Show Possible Inconsistencies toggle command is switched on. If an inconsistency possibly exists and if the Show Possible Inconsistencies toggle is switched off, then the semantics check returns a result indicating the

 © 2006 Altova GmbH

existence of apparent inconsistency. The possible inconsistencies can be viewed by switching on the Show Possible Inconsistencies toggle. Only if no possible inconsistency is detected does the semantics check return a positive result.

Also see Errors Window.

5.7 Tools Menu

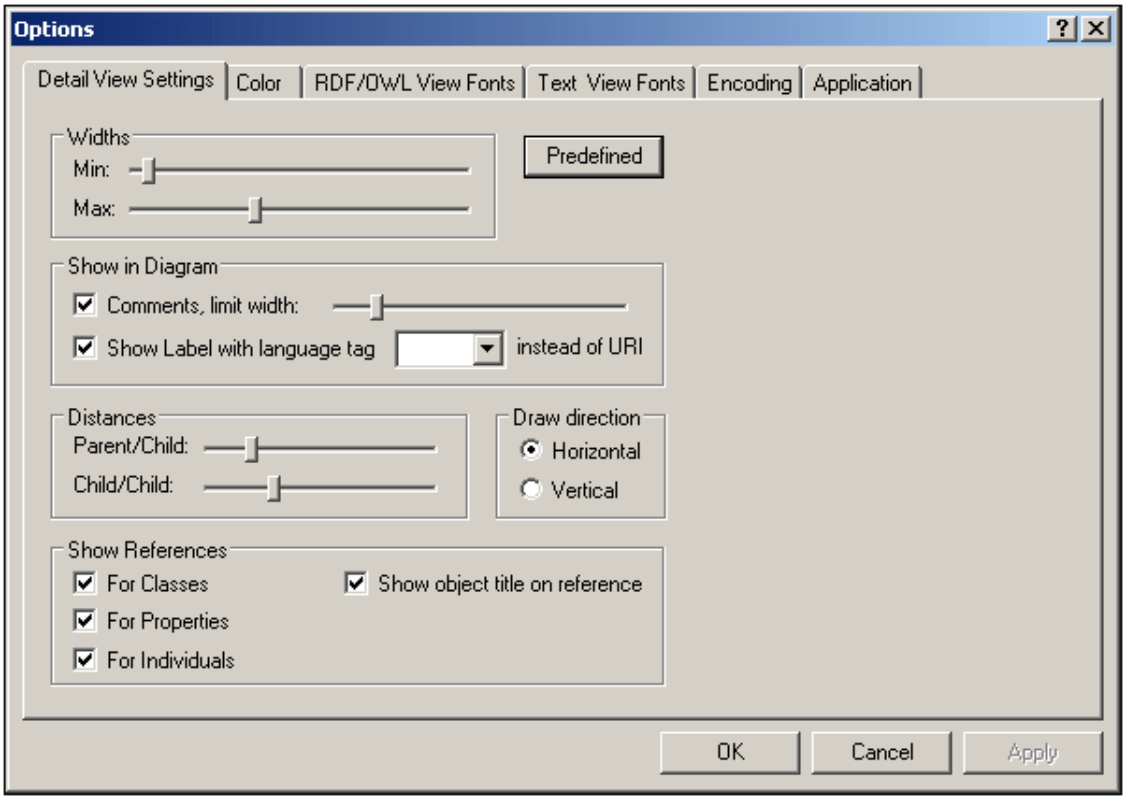
The commands in the **Tools menu** enable you to customize the application (Options menu item) and set up the active document's namespaces, URIref prefixes, URIref serialization, and base URI. These commands are described in detail in the subsections of this section:

- Options: describes the options that can be set in the various tabs of the Options dialog.
- Namespace Imports for RDF: describes how the Namespace Imports dialog can be used to import namespaces and thereby make available resources described in an ontology for insertion in an RDF document.
- URIref Prefixes and Namespaces: describes the URIref Prefixes dialog and the Expand URIref Prefixes feature.
- Base URI: describes how to define a base URI for a document using the Base URI dialog.

© 2006 Altova GmbH

5.7.1 Options

Clicking the **Options** menu item pops up the Options dialog (*screenshot below*), in which you can set options for the application.



The **Detail View Settings** tab enables you to make the following settings for Detail View.

- In the Draw Direction pane, set the direction in which the Detail View of an item is drawn: from left to right (Horizontal), or from top to bottom (Vertical).
- In the Widths pane, set the minimum and maximum width of object boxes.
- In the Distances pane, set the distance between parent and child objects, and between child objects.
- In the Show in Diagram pane, set (i) whether comments are shown, and set their widths; (ii) whether labels are shown instead of URIs.
- In the Show References pane, set whether references to classes, properties, and individuals are displayed.

To revert to the original Altova-defined settings click the Predefined button.

Further tabs in the Options dialog enable you to customize SemanticWorks as follows:

- The **Color** tab allows you to design a background color for the Detail View. You can set the background to be a solid color or a gradient.
- In the **RDF/OWL View Fonts** and **Text View Fonts** tabs you can set the font face, font size, font style, and font color for various items in RDF/OWL View and for text in Text View.
- In the **Encoding** tab, you can select the default encoding for XML and non-XML files.
- In the **Application** tab, you can select whether the SemanticWorks application logo should be displayed when the program starts and whether it should be printed when a document is printed from within SemanticWorks. You can also select whether imports should be resolved and, optionally, validated when a document is opened, or not.

© 2006 Altova GmbH

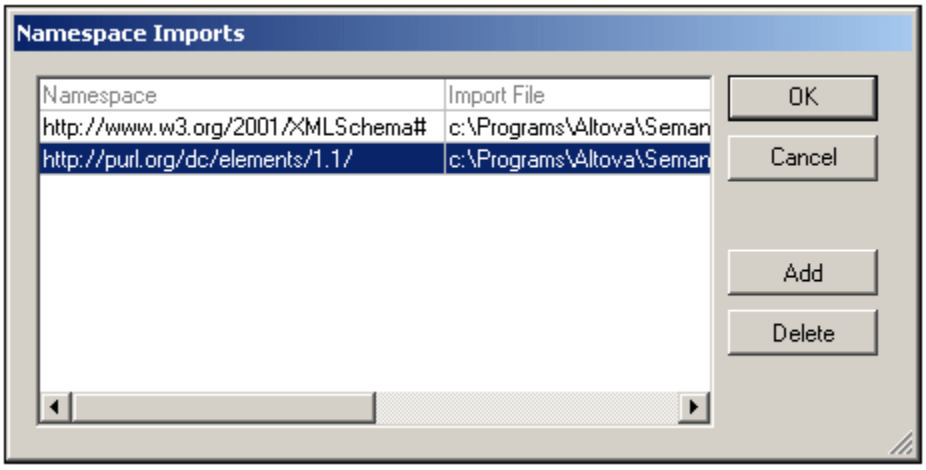
5.7.2 Namespace Imports for RDF

When creating an RDF or RDFS document, it is convenient to be able to insert resources in RDF statements by selecting the required resources from a list. This is especially useful if a single resource is to be inserted multiple times in a document.

As an example of such use consider the creation of an RDF document that contains metadata for a large number of web pages. Each web page resource is described by the same set of property resources. If the property resources are described in an ontology, then SemanticWorks can access these resources so that they can be displayed in the SemanticWorks interface and be inserted in RDF statements. SemanticWorks does this using the Namespace Imports mechanism.

The Namespace Imports mechanism works by importing, into the RDF or RDFS document, the namespace URIs of the resources to be referenced. The imported namespace URIs must be the same as those used to define the required resources in an ontology. Once a namespace URI is imported, ontology resources associated with this URI are made available to SemanticWorks for insertion in the active document. The **Namespace Imports for RDF** command is the SemanticWorks feature that enables you to import the required namespaces. The feature works as follows:

1. Select the Namespace Imports for RDF command in order to display the Namespace Imports dialog (*screenshot below*).



2. In the Namespace column enter the first namespace to be imported, say, `http//purl.org/dc/elements/1.1/`, which is the DC namespace declared, say, in an ontology document `DCOntology.rdf`.
3. In the Import File column, enter the location of the ontology document `DCOntology.rdf`. Note that you should give the absolute path for the ontology document.
4. If the ontology file defines XML Schema datatypes, you will need to import the XML Schema namespace (`http//www.w3.org/2001/XMLSchema#`), too. Enter the XML Schema namespace (`http://www.w3.org/2001/XMLSchema#`) in the Namespace column and the location of the ontology file in the Import File column.
5. Click OK to complete.

In the procedure outlined above, you have imported two namespace URIs. After declaring these namespaces in your RDF document (see URIref Prefixes and Namespaces for a description of how to do this), resources from the ontology file from which the namespaces have been imported are listed as resources (in RDF documents) or instances (in RDFS documents); and (ii) are available in the Detail View of resources (or instances) , for insertion in the RDF or RDFS

document.

For a detailed description of usage in practice, see the tutorial section RDF Documents.

Note: Resources that are displayed in the Resources or Instances Overview as a result of the namespace imports (and not as a result of being physically entered in the document) are available only for insertion. They should be regarded as abstract resources (available for instantiation) and distinct from the resources actually contained in the document.

Namespace imports and `owl:imports`

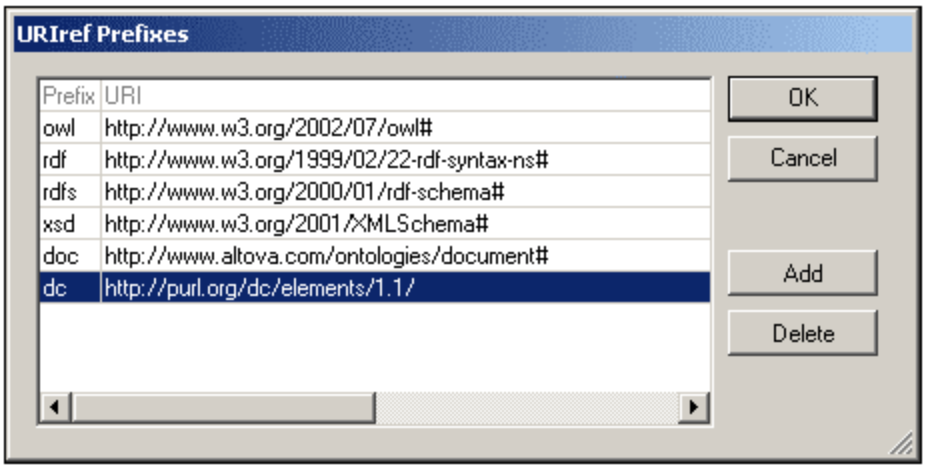
The Namespace Import feature can also be used with the `owl:imports` ontology property. In order for the namespace import to work, the base URI of the OWL files, which is specified using the `xml:base` attribute, must indicate the original URI—that is, the URI used in the `owl:imports` ontology property. (The value of `xml:base` should not point to the location of the file being mapped for the namespace import.)

© 2006 Altova GmbH

5.7.3 URIref Prefixes and Namespaces

This section describes the menu items: **URIref Prefixes** and **Expand URIref Prefixes**. Also see the related topic, Namespace Imports for RDF.

URIref Prefixes
Clicking the URIref Prefixes menu item pops up the URIref Prefixes dialog (*screenshot below*), in which all the namespaces declared for the active ontology are displayed. (The RDF, RDF Schema, and OWL namespaces are declared by default for new documents.) Via the dialog, you can (i) add namespaces to the ontology and bind these namespaces to customized prefixes, (ii) edit existing namespaces and prefixes, and (iii) delete a selected namespace.



To add a namespace or delete the selected namespace, use the Add and Delete buttons respectively. To edit a namespace or prefix, place the cursor in the appropriate field and edit using the keyboard.

Expand URIref Prefixes
After prefixes for namespaces have been assigned (see URIref Prefixes above), a resource can be defined or referenced in the RDF/XML serialization using either: (i) the prefix and local name, or (ii) the expanded URIref. To use the expanded URIref in the serialization, toggle the **Expand URIref Prefixes** command on. All URIrefs entered in RDF/OWL View after this, that have a prefix, will be serialized in the RDF/XML notation to the expanded URIref form if that prefix has been declared. If the prefix does not exist, then the URIref is serialized exactly as entered, that is, the prefix is read as the scheme part of a URI. The RDF/OWL View itself always displays the URIref as entered; URIrefs are not expanded in RDF/OWL View.

In actuality, what happens when this command is toggled on, is that every input is solely understood as a URI (absolute or relative) and is **not** understood as prefix-and-local-name. That is, prefixes that would have been expanded to a URI part when the option is activated are now simply used as the scheme part of the URI that is entered. Note that all relative URIs will be resolved against the document's global base URI.

5.7.4 Base URI

The Base URI command pops up the Base URI dialog (*screenshot below*), in which you enter the base URI you wish the active document to have. The base URI is useful for resolving relative paths.

The base URI you submit is entered as the value of the `xml:base` attribute of the `rdf:RDF` element of the document. For example:

```
<rdf:RDF xml:base="http://www.altova.com/ontologies/Documents.rdf"/>
```

Note: By default, the base URI is the URI of the document. This default URI is not explicitly serialized in the form of a value for the `xml:base` attribute. Even when the URI of the document is entered in the Base URI dialog, it is not serialized in the RDF/XML. Only a URI that is not the URI of the document is serialized.

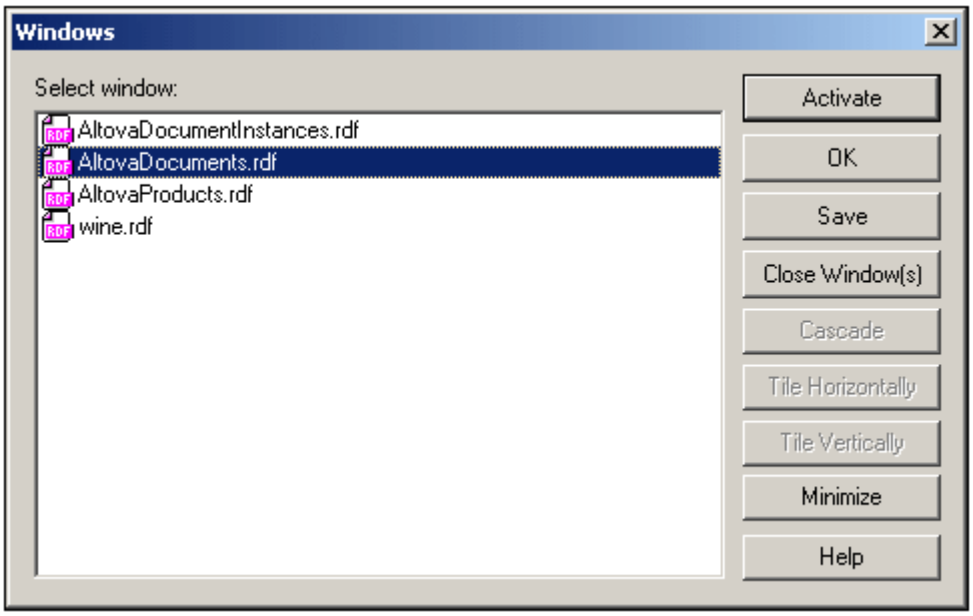
© 2006 Altova GmbH

5.8 Window Menu

The **Window menu** has commands to specify how SemanticWorks windows should be displayed in the GUI (cascaded, tiled, or maximized). To maximize a window, click the maximize button of that window.

Additionally, all currently open document windows are listed in this menu by document name, with the active window being checked. To make another window active, click the name of the window you wish to make active.

Windows dialog
At the bottom of the list of open windows is an entry for the Windows dialog. Clicking this entry opens the Windows dialog, which displays a list of all open windows and provides commands that can be applied to the selected window/s. (A window is selected by clicking on its name.)



The Cascade and Tile options are available only when more than one window is selected. The Activate option is enabled only when a single window is selected.

Warning: To exit the Windows dialog, click OK; do **not** click the Close Window(s) button. The Close Window(s) button closes the window/s currently selected in the Windows dialog.

5.9 Help Menu

The **Help menu** contains an onscreen version of this documentation, registration information, relevant Internet hyperlinks, and information about your version of SemanticWorks.

The online documentation can be opened in one of three browsing modes: Table of Contents, Index, and Search. If you are online when using SemanticWorks, you can also register your product with Altova, order Altova products from the Altova website, and access Altova Support databases. The **About SemanticWorks...** command gives you SemanticWorks product information.

© 2006 Altova GmbH

5.10 Usage Issues

The following usage issue should be noted:

Cardinality of property restrictions
When entering property restrictions in `intersectionOf` statements of an ontology, the `mincardinality` and `maxcardinality` cannot be entered on a single restriction; they must be entered on two separate restrictions for that property.

Chapter 6

Conformance

6 Conformance

SemanticWorks 2006 conforms to the W3C specifications listed in the W3C's RDF Overview and OWL Overview documents. The respective suites of specifications are as listed below.

RDF specifications

- RDF Primer
- RDF Concepts and Abstract Syntax
- RDF/XML Syntax
- RDF Semantics
- RDF Vocabulary Description Language 1.0 (RDF Schema)

OWL specifications

- OWL Web Ontology Language Guide
- OWL Semantics and Abstract Syntax
- OWL Web Ontology Language Reference

Implementation-specific information
The following implementation-specific information should be noted:

The syntax check of the document
Checks if the document is well-formed RDF. The Syntax Check command of SemanticWorks checks whether the document can be transformed into OWL Abstract Syntax following the rules given in the *OWL Web Ontology Language Semantics and Abstract Syntax* document, *Section 4: Mapping to RDF Graphs*. The document is said to be well-formed if it satisfies the definitions for an "OWL Lite ontology in RDF graph form" or an "OWL DL ontology in RDF graph form", respectively for OWL Lite and OWL DL documents, as described at the end of Section 4.

The semantic check of the document
The Semantic Check command of SemanticWorks checks whether the document follows the rules given in the *OWL Web Ontology Language Semantics and Abstract Syntax* document, *Section 5: RDF-Compatible Model-Theoretic Semantics*. The semantic engine in SemanticWorks executes the checks solely on the existing statements. It is a partial consistency checker.

© 2006 Altova GmbH

Chapter 7

License Information

7 License Information

This section contains:

- Information about the distribution of this software product
- Information about the copyrights related to this software product
- The End User License Agreement governing the use of this software product

Please read this information carefully. It is binding upon you since you agreed to these terms when you installed this software product.

© 2006 Altova GmbH

7.1 Electronic Software Distribution

This product is available through electronic software distribution, a distribution method that provides the following unique benefits:

- You can evaluate the software free-of-charge before making a purchasing decision.
- Once you decide to buy the software, you can place your order online at the Altova website and immediately get a fully licensed product within minutes.
- When you place an online order, you always get the latest version of our software.
- The product package includes a comprehensive integrated onscreen help system. The latest version of the user manual is available at www.altova.com (i) in HTML format for online browsing, and (ii) in PDF format for download (and to print if you prefer to have the documentation on paper).

30-day evaluation period
After downloading this product, you can evaluate it for a period of up to 30 days free of charge. About 20 days into this evaluation period, the software will start to remind you that it has not yet been licensed. The reminder message will be displayed once each time you start the application. If you would like to continue using the program after the 30-day evaluation period, you have to purchase an End User License Agreement, which is delivered in the form of a key-code that you enter into the Registration dialog to unlock the product. You can purchase your license at the online shop at the Altova website.

Distributing the product
If you wish to share the product with others, please make sure that you distribute only the installation program, which is a convenient package that will install the application together with all sample files and the onscreen help. Any person that receives the product from you is also automatically entitled to a 30-day evaluation period. After the expiration of this period, any other user must also purchase a license in order to be able to continue using the product.

For further details, please refer to the End User License Agreement at the end of this section.

7.2 License Metering

Your Altova product has a built-in license metering module that helps you avoid any unintentional violation of the End User License Agreement. Your product is licensed either as a single-user or multi-user installation, and the license-metering module makes sure that no more than the licensed number of users use the application concurrently.

This license-metering technology uses your local area network (LAN) to communicate between instances of the application running on different computers.

Single license
When the application starts up, it sends a short broadcast datagram to find any other instance of the product running on another computer in the same network segment. If it doesn't get any response, it will open a port for listening to other instances of the application. Other than that, it will not attempt to communicate over a network. If you are not connected to a LAN, or are using dial-up connections to connect to the Internet, the application will not generate any network traffic at all.

Multi license
If more than one instance of the application is used within the same LAN, these instances will briefly communicate with each other on startup. These instances exchange key-codes in order to ensure that the number of concurrent licenses purchased is not accidentally violated. This is the same kind of license metering technology that is common in the Unix world and with a number of database development tools. It allows Altova customers to purchase reasonably-priced concurrent-use multi-user licenses..

Please note that your Altova product at no time attempts to send any information out of your LAN or over the Internet. We have also designed the applications so that they send few and small network packets so as to not put a burden on your network. The TCP/IP ports (2799) used by your Altova product are officially registered with the IANA (*see* http://www.isi.edu/in-notes/iana/assignments/port-numbers *for details*) and our license-metering module is tested and proven technology.

If you are using a firewall, you may notice communications on port 2799 between the computers that are running Altova products. You are, of course, free to block such traffic between different groups in your organization, as long as you can ensure by other means, that your license agreement is not violated.

You will also notice that, if you are online, your Altova product contains many useful functions; these are unrelated to the license-metering technology.

© 2006 Altova GmbH

7.3 Copyright

All title and copyrights in this software product (including but not limited to images, photographs, animations, video, audio, music, text, and applets incorporated in the product), in the accompanying printed materials, and in any copies of these printed materials are owned by Altova GmbH or the respective supplier. This software product is protected by copyright laws and international treaty provisions.

- This software product ©1998-2006 Altova GmbH. All rights reserved.
- The Sentry Spelling-Checker Engine © 2000 Wintertree Software Inc.
- STLport © 1999, 2000 Boris Fomitchev, © 1994 Hewlett-Packard Company, © 1996, 1997 Silicon Graphics Computer Systems, Inc, © 1997 Moscow Center for SPARC Technology.
- Scintilla © 1998–2002 Neil Hodgson <neilh@scintilla.org>.
- "ANTLR Copyright © 1989-2005 by Terence Parr (www.antlr.org)"

All other names or trademarks are the property of their respective owners.

7.4 Altova End User License Agreement

THIS IS A LEGAL DOCUMENT -- RETAIN FOR YOUR RECORDS

ALTOVA® END USER LICENSE AGREEMENT

Licensor:

Altova GmbH
Rudolfsplatz 13a/9
A-1010 Wien
Austria

Important - Read Carefully. Notice to User:

This End User License Agreement ("Software License Agreement") is a legal document between you and Altova GmbH ("Altova"). It is important that you read this document before using the Altova-provided software ("Software") and any accompanying documentation, including, without limitation printed materials, 'online' files, or electronic documentation ("Documentation"). By clicking the "I accept" and "Next" buttons below, or by installing, or otherwise using the Software, you agree to be bound by the terms of this Software License Agreement as well as the Altova Privacy Policy ("Privacy Policy") including, without limitation, the warranty disclaimers, limitation of liability, data use and termination provisions below, whether or not you decide to purchase the Software. You agree that this agreement is enforceable like any written agreement negotiated and signed by you. If you do not agree, you are not licensed to use the Software, and you must destroy any downloaded copies of the Software in your possession or control. Please go to our Web site at http://www.altova.com/eula to download and print a copy of this Software License Agreement for your files and http://www.altova.com/privacy to review the privacy policy.

1. SOFTWARE LICENSE

(a) **License Grant.** Upon your acceptance of this Software License Agreement Altova grants you a non-exclusive, non-transferable (except as provided below), limited license to install and use a copy of the Software on your compatible computer, up to the Permitted Number of computers. The Permitted Number of computers shall be delineated at such time as you elect to purchase the Software. During the evaluation period, hereinafter defined, only a single user may install and use the software on one computer. If you have licensed the Software as part of a suite of Altova software products (collectively, the "Suite") and have not installed each product individually, then the Software License Agreement governs your use of all of the software included in the Suite. If you have licensed SchemaAgent, then the terms and conditions of this Software License Agreement apply to your use of the SchemaAgent server software ("SchemaAgent Server") included therein, as applicable and you are licensed to use SchemaAgent Server solely in connection with your use of Altova Software and solely for the purposes described in the accompanying documentation. In addition, if you have licensed XMLSpy Enterprise Edition or MapForce Enterprise Edition, or UModel, your license to install and use a copy of the Software as provided herein permits you to generate source code based on (i) Altova Library modules that are included in the Software (such generated code hereinafter referred to as the "Restricted Source Code") and (ii) schemas or mappings that you create or provide (such code as may be generated from your schema or mapping source materials hereinafter referred to as the "Unrestricted Source Code"). In addition to the rights granted herein, Altova grants you a non-exclusive, non-transferable, limited license to compile into executable form the complete generated code comprised of the combination of the Restricted Source Code and the Unrestricted Source Code, and to use, copy, distribute or license that executable. You may not distribute or redistribute, sublicense, sell, or transfer to a third party the Restricted Source Code, unless said third party already has a license to the Restricted Source Code through their separate license agreement with Altova or other agreement with

 © 2006 Altova GmbH

Altova. Altova reserves all other rights in and to the Software. With respect to the feature(s) of UModel that permit reverse-engineering of your own source code or other source code that you have lawfully obtained, such use by you does not constitute a violation of this Agreement. Except as otherwise permitted in Section 1(h) reverse engineering of the Software is strictly prohibited as further detailed therein.

(b) **Server Use.** You may install one copy of the Software on your computer file server for the purpose of downloading and installing the Software onto other computers within your internal network up to the Permitted Number of computers. If you have licensed .SchemaAgent, then you may install SchemaAgent Server on any server computer or workstation and use it in connection with your Software. No other network use is permitted, including without limitation using the Software either directly or through commands, data or instructions from or to a computer not part of your internal network, for Internet or Web-hosting services or by any user not licensed to use this copy of the Software through a valid license from Altova.

(c) **Concurrent Use**. If you have licensed a "Concurrent-User" version of the Software, you may install the Software on any compatible computers, up to ten (10) times the Permitted Number of users, provided that only the Permitted Number of users actually use the Software at the same time. The Permitted Number of concurrent users shall be delineated at such time as you elect to purchase the Software licenses.

(d) **Backup and Archival Copies.** You may make one backup and one archival copy of the Software, provided your backup and archival copies are not installed or used on any computer and further provided that all such copies shall bear the original and unmodified copyright, patent and other intellectual property markings that appear on or in the Software. You may not transfer the rights to a backup or archival copy unless you transfer all rights in the Software as provided under Section 3.

(e) **Home Use.** You, as the primary user of the computer on which the Software is installed, may also install the Software on one of your home computers for your use. However, the Software may not be used on your home computer at the same time as the Software is being used on the primary computer.

(f) **Key Codes, Upgrades and Updates.** Prior to your purchase and as part of the registration for the thirty (30) -day evaluation period, as applicable, you will receive an evaluation key code. You will receive a purchase key code when you elect to purchase the Software. The purchase key code will enable you to activate the Software beyond the initial evaluation period. You may not re-license, reproduce or distribute any key code except with the express written permission of Altova. If the Software that you have licensed is an upgrade or an update, then the update replaces all or part of the Software previously licensed. The update or upgrade and the associated license keys does not constitute the granting of a second license to the Software in that you may not use the upgrade or update in addition to the Software that it is replacing. You agree that use of the upgrade of update terminates your license to use the Software or portion thereof replaced.

(g) **Title.** Title to the Software is not transferred to you. Ownership of all copies of the Software and of copies made by you is vested in Altova, subject to the rights of use granted to you in this Software License Agreement. As between you and Altova, documents, files, stylesheets, generated program code (including the Unrestricted Source Code) and schemas that are authored or created by you via your utilization of the Software, in accordance with its Documentation and the terms of this Software License Agreement, are your property.

(h) **Reverse Engineering.** Except and to the limited extent as may be otherwise specifically provided by applicable law in the European Union, you may not reverse engineer, decompile, disassemble or otherwise attempt to discover the source code, underlying ideas, underlying user interface techniques or algorithms of the Software by any means whatsoever, directly or indirectly, or disclose any of the foregoing, except to the extent you may be expressly permitted to decompile under applicable law in the European Union, if it is essential to do so in order to achieve operability of the Software with another software program, and you have first requested Altova to provide the information necessary to achieve such operability and Altova has not made such information available. Altova has the right to impose reasonable conditions and to request a reasonable fee before providing such information. Any information supplied by Altova or obtained by you, as permitted hereunder, may only be used by you for the purpose

described herein and may not be disclosed to any third party or used to create any software which is substantially similar to the expression of the Software. Requests for information from users in the European Union with respect to the above should be directed to the Altova Customer Support Department.

(i) **Other Restrictions.** You may not loan, rent, lease, sublicense, distribute or otherwise transfer all or any portion of the Software to third parties except to the limited extent set forth in Section 3 or otherwise expressly provided. You may not copy the Software except as expressly set forth above, and any copies that you are permitted to make pursuant to this Software License Agreement must contain the same copyright, patent and other intellectual property markings that appear on or in the Software. You may not modify, adapt or translate the Software. You may not, directly or indirectly, encumber or suffer to exist any lien or security interest on the Software; knowingly take any action that would cause the Software to be placed in the public domain; or use the Software in any computer environment not specified in this Software License Agreement. You will comply with applicable law and Altova's instructions regarding the use of the Software. You agree to notify your employees and agents who may have access to the Software of the restrictions contained in this Software License Agreement and to ensure their compliance with these restrictions. you agree that you are solely responsible for the accuracy and adequacy of the software for your intended use and you will indemnify and HOLD harmless ALTOVA from any 3rd party suit to the extent based upon the accuracy and adequacy of the software in your use. without limitation, The Software is not intended for use in the operation of nuclear facilities, aircraft navigation, communication systems or air traffic control equipment, where the failure of the Software could lead to death, personal injury or severe physical or environmental damage.

2. INTELLECTUAL PROPERTY RIGHTS

Acknowledgement of Altova's Rights. You acknowledge that the Software and any copies that you are authorized by Altova to make are the intellectual property of and are owned by Altova and its suppliers. The structure, organization and code of the Software are the valuable trade secrets and confidential information of Altova and its suppliers. The Software is protected by copyright, including without limitation by United States Copyright Law, international treaty provisions and applicable laws in the country in which it is being used. You acknowledge that Altova retains the ownership of all patents, copyrights, trade secrets, trademarks and other intellectual property rights pertaining to the Software, and that Altova's ownership rights extend to any images, photographs, animations, videos, audio, music, text and "applets" incorporated into the Software and all accompanying printed materials. You will take no actions which adversely affect Altova's intellectual property rights in the Software. Trademarks shall be used in accordance with accepted trademark practice, including identification of trademark owners' names. Trademarks may only be used to identify printed output produced by the Software, and such use of any trademark does not give you any right of ownership in that trademark. XMLSpy, Authentic, StyleVision, MapForce, Markup Your Mind, Axad, Nanonull, and Altova are trademarks of Altova GmbH (registered in numerous countries). Unicode and the Unicode Logo are trademarks of Unicode, Inc. Windows, Windows 95, Windows 98, Windows NT, Windows 2000 and Windows XP are trademarks of Microsoft. W3C, CSS, DOM, MathML, RDF, XHTML, XML and XSL are trademarks (registered in numerous countries) of the World Wide Web Consortium (W3C); marks of the W3C are registered and held by its host institutions, MIT, INRIA and Keio. Except as expressly stated above, this Software License Agreement does not grant you any intellectual property rights in the Software. Notifications of claimed copyright infringement should be sent to Altova's copyright agent as further provided on the Altova Web Site.

3. LIMITED TRANSFER RIGHTS

Notwithstanding the foregoing, you may transfer all your rights to use the Software to another person or legal entity provided that: (a) you also transfer each of this Software License Agreement, the Software and all other software or hardware bundled or pre-installed with the Software, including all copies, updates and prior versions, and all copies of font software converted into other formats, to such person or entity; (b) you retain no copies, including

backups and copies stored on a computer; (c) the receiving party secures a personalized key code from Altova; and (d) the receiving party accepts the terms and conditions of this Software License Agreement and any other terms and conditions upon which you legally purchased a license to the Software. Notwithstanding the foregoing, you may not transfer education, pre-release, or not-for-resale copies of the Software.

4. PRE-RELEASE AND EVALUATION PRODUCT ADDITIONAL TERMS

If the product you have received with this license is pre-commercial release or beta Software ("Pre-release Software"), then this Section applies. In addition, this section applies to all evaluation and/or demonstration copies of Altova software ("Evaluation Software") and continues in effect until you purchase a license. To the extent that any provision in this section is in conflict with any other term or condition in this Software License Agreement, this section shall supersede such other term(s) and condition(s) with respect to the Pre-release and/or Evaluation Software, but only to the extent necessary to resolve the conflict. You acknowledge that the Pre-release Software is a pre-release version, does not represent final product from Altova, and may contain bugs, errors and other problems that could cause system or other failures and data loss. CONSEQUENTLY, THE PRE-RELEASE AND/OR EVALUATION SOFTWARE IS PROVIDED TO YOU **"AS-IS" WITH NO WARRANTIES FOR USE OR PERFORMANCE,** AND ALTOVA DISCLAIMS ANY WARRANTY OR LIABILITY OBLIGATIONS TO YOU OF ANY KIND, WHETHER EXPRESS OR IMPLIED. WHERE LEGALLY LIABILITY CANNOT BE EXCLUDED FOR PRE-RELEASE AND/OR EVALUATION SOFTWARE, BUT IT MAY BE LIMITED, ALTOVA'S LIABILITY AND THAT OF ITS SUPPLIERS SHALL BE LIMITED TO THE SUM OF FIFTY DOLLARS (USD $50) IN TOTAL. If the Evaluation Software has a time-out feature, then the software will cease operation after the conclusion of the designated evaluation period. Upon such expiration date, your license will expire unless otherwise extended. Access to any files created with the Evaluation Software is entirely at your risk. You acknowledge that Altova has not promised or guaranteed to you that Pre-release Software will be announced or made available to anyone in the future, that Altova has no express or implied obligation to you to announce or introduce the Pre-release Software, and that Altova may not introduce a product similar to or compatible with the Pre-release Software. Accordingly, you acknowledge that any research or development that you perform regarding the Pre-release Software or any product associated with the Pre-release Software is done entirely at your own risk. During the term of this Software License Agreement, if requested by Altova, you will provide feedback to Altova regarding testing and use of the Pre-release Software, including error or bug reports. If you have been provided the Pre-release Software pursuant to a separate written agreement, your use of the Software is governed by such agreement. You may not sublicense, lease, loan, rent, distribute or otherwise transfer the Pre-release Software. Upon receipt of a later unreleased version of the Pre-release Software or release by Altova of a publicly released commercial version of the Software, whether as a stand-alone product or as part of a larger product, you agree to return or destroy all earlier Pre-release Software received from Altova and to abide by the terms of the license agreement for any such later versions of the Pre-release Software.

5. LIMITED WARRANTY AND LIMITATION OF LIABILITY

(a) **Limited Warranty and Customer Remedies.** Altova warrants to the person or entity that first purchases a license for use of the Software pursuant to the terms of this Software License Agreement that (i) the Software will perform substantially in accordance with any accompanying Documentation for a period of ninety (90) days from the date of receipt, and (ii) any support services provided by Altova shall be substantially as described in Section 6 of this agreement. Some states and jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you. To the extent allowed by applicable law, implied warranties on the Software, if any, are limited to ninety (90) days. Altova's and its suppliers' entire liability and your exclusive remedy shall be, at Altova's option, either (i) return of the price paid, if any, or (ii) repair or replacement of the Software that does not meet Altova's Limited Warranty and which is returned to Altova with a copy of your receipt. This Limited Warranty is void if failure of the Software has resulted from accident, abuse, misapplication,

abnormal use, Trojan horse, virus, or any other malicious external code. Any replacement Software will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer. This limited warranty does not apply to Evaluation and/or Pre-release Software.

(b) **No Other Warranties and Disclaimer.** THE FOREGOING LIMITED WARRANTY AND REMEDIES STATE THE SOLE AND EXCLUSIVE REMEDIES FOR ALTOVA OR ITS SUPPLIER'S BREACH OF WARRANTY. ALTOVA AND ITS SUPPLIERS DO NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULTS YOU MAY OBTAIN BY USING THE SOFTWARE. EXCEPT FOR THE FOREGOING LIMITED WARRANTY, AND FOR ANY WARRANTY, CONDITION, REPRESENTATION OR TERM TO THE EXTENT WHICH THE SAME CANNOT OR MAY NOT BE EXCLUDED OR LIMITED BY LAW APPLICABLE TO YOU IN YOUR JURISDICTION, ALTOVA AND ITS SUPPLIERS MAKE NO WARRANTIES, CONDITIONS, REPRESENTATIONS OR TERMS, EXPRESS OR IMPLIED, WHETHER BY STATUTE, COMMON LAW, CUSTOM, USAGE OR OTHERWISE AS TO ANY OTHER MATTERS. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, ALTOVA AND ITS SUPPLIERS DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, INFORMATIONAL CONTENT OR ACCURACY, QUIET ENJOYMENT, TITLE AND NON-INFRINGEMENT, WITH REGARD TO THE SOFTWARE, AND THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES. THIS LIMITED WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHERS, WHICH VARY FROM STATE/JURISDICTION TO STATE/JURISDICTION.

(c) **Limitation Of Liability.** TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW EVEN IF A REMEDY FAILS ITS ESSENTIAL PURPOSE, IN NO EVENT SHALL ALTOVA OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE OR THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, EVEN IF ALTOVA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, ALTOVA'S ENTIRE LIABILITY UNDER ANY PROVISION OF THIS SOFTWARE LICENSE AGREEMENT SHALL BE LIMITED TO THE AMOUNT ACTUALLY PAID BY YOU FOR THE SOFTWARE PRODUCT. Because some states and jurisdictions do not allow the exclusion or limitation of liability, the above limitation may not apply to you. In such states and jurisdictions, Altova's liability shall be limited to the greatest extent permitted by law and the limitations or exclusions of warranties and liability contained herein do not prejudice applicable statutory consumer rights of person acquiring goods otherwise than in the course of business. The disclaimer and limited liability above are fundamental to this Software License Agreement between Altova and you.

(d) **Infringement Claims.** Altova will indemnify and hold you harmless and will defend or settle any claim, suit or proceeding brought against you by a third party that is based upon a claim that the content contained in the Software infringes a copyright or violates an intellectual or proprietary right protected by United States or European Union law ("Claim"), but only to the extent the Claim arises directly out of the use of the Software and subject to the limitations set forth in Section 5 of this Agreement except as otherwise expressly provided. You must notify Altova in writing of any Claim within ten (10) business days after you first receive notice of the Claim, and you shall provide to Altova at no cost with such assistance and cooperation as Altova may reasonably request from time to time in connection with the defense of the Claim. Altova shall have sole control over any Claim (including, without limitation, the selection of counsel and the right to settle on your behalf on any terms Altova deems desirable in the sole exercise of its discretion). You may, at your sole cost, retain separate counsel and participate in the defense or settlement negotiations. Altova shall pay actual damages, costs, and attorney fees awarded against you (or payable by you pursuant to a settlement agreement) in connection with a Claim

to the extent such direct damages and costs are not reimbursed to you by insurance or a third party, to an aggregate maximum equal to the purchase price of the Software. If the Software or its use becomes the subject of a Claim or its use is enjoined, or if in the opinion of Altova's legal counsel the Software is likely to become the subject of a Claim, Altova shall attempt to resolve the Claim by using commercially reasonable efforts to modify the Software or obtain a license to continue using the Software. If in the opinion of Altova's legal counsel the Claim, the injunction or potential Claim cannot be resolved through reasonable modification or licensing, Altova, at its own election, may terminate this Software License Agreement without penalty, and will refund to you on a pro rata basis any fees paid in advance by you to Altova. THE FOREGOING CONSTITUTES ALTOVA'S SOLE AND EXCLUSIVE LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT. This indemnity does not apply to infringements that would not be such, except for customer-supplied elements.

6. SUPPORT AND MAINTENANCE

Altova offers multiple optional "Support & Maintenance Package(s)" ("SMP") for the version of Software product edition that you have licensed, which you may elect to purchase in addition to your Software license. The Support Period, hereinafter defined, covered by such SMP shall be delineated at such time as you elect to purchase a SMP. Your rights with respect to support and maintenance as well as your upgrade eligibility depend on your decision to purchase SMP and the level of SMP that you have purchased:

(a) If you have not purchased SMP, you will receive the Software AS IS and will not receive any maintenance releases or updates. However, Altova, at its option and in its sole discretion on a case by case basis, may decide to offer maintenance releases to you as a courtesy, but these maintenance releases will not include any new features in excess of the feature set at the time of your purchase of the Software. In addition, Altova will provide free technical support to you for thirty (30) days after the date of your purchase (the "Support Period" for the purposes of this paragraph a), and Altova, in its sole discretion on a case by case basis, may also provide free courtesy technical support during your thirty (30)-day evaluation period. Technical support is provided via a Web-based support form only, and there is no guaranteed response time.

(b) If you have purchased SMP, then solely for the duration of its delineated Support Period, **you are eligible to receive the version of the Software edition** that you have licensed and all maintenance releases and updates for that edition that are released during your Support Period. For the duration of your SMP's Support Period, you will also be eligible to receive upgrades to the comparable edition of the next version of the Software that succeeds the Software edition that you have licensed for applicable upgrades released during your Support Period. The specific upgrade edition that you are eligible to receive based on your Support Period is further detailed in the SMP that you have purchased. Software that is introduced as separate product is not included in SMP. Maintenance releases, updates and upgrades may or may not include additional features. In addition, Altova will provide Priority Technical Support to you for the duration of the Support Period. Priority Technical Support is provided via a Web-based support form only, and Altova will make commercially reasonable efforts to respond via e-mail to all requests within forty-eight (48) hours during Altova's business hours (MO-FR, 8am UTC – 10pm UTC, Austrian and US holidays excluded) and to make reasonable efforts to provide work-arounds to errors reported in the Software.

During the Support Period you may also report any Software problem or error to Altova. If Altova determines that a reported reproducible material error in the Software exists and significantly impairs the usability and utility of the Software, Altova agrees to use reasonable commercial efforts to correct or provide a usable work-around solution in an upcoming maintenance release or update, which is made available at certain times at Altova's sole discretion.

If Altova, in its discretion, requests written verification of an error or malfunction discovered by you or requests supporting example files that exhibit the Software problem, you shall promptly provide such verification or files, by email, telecopy, or overnight mail, setting forth in reasonable detail the respects in which the Software fails to perform. You shall use reasonable efforts to cooperate in diagnosis or study of errors. Altova may include error corrections in

maintenance releases, updates, or new major releases of the Software. Altova is not obligated to fix errors that are immaterial. Immaterial errors are those that do not significantly impact use of the Software. Whether or not you have purchased the Support & Maintenance Package, technical support only covers issues or questions resulting directly out of the operation of the Software and Altova will not provide you with generic consultation, assistance, or advice under any circumstances.
Updating Software may require the updating of software not covered by this Software License Agreement before installation. Updates of the operating system and application software not specifically covered by this Software License Agreement are your responsibility and will not be provided by Altova under this Software License Agreement. Altova's obligations under this Section 6 are contingent upon your proper use of the Software and your compliance with the terms and conditions of this Software License Agreement at all times. Altova shall be under no obligation to provide the above technical support if, in Altova's opinion, the Software has failed due to the following conditions: (i) damage caused by the relocation of the software to another location or CPU; (ii) alterations, modifications or attempts to change the Software without Altova's written approval; (iii) causes external to the Software, such as natural disasters, the failure or fluctuation of electrical power, or computer equipment failure; (iv) your failure to maintain the Software at Altova's specified release level; or (v) use of the Software with other software without Altova's prior written approval. It will be your sole responsibility to: (i) comply with all Altova-specified operating and troubleshooting procedures and then notify Altova immediately of Software malfunction and provide Altova with complete information thereof; (ii) provide for the security of your confidential information; (iii) establish and maintain backup systems and procedures necessary to reconstruct lost or altered files, data or programs.

7. SOFTWARE ACTIVATION, UPDATES AND LICENSE METERING

(a) **License Metering**. Altova has a built-in license metering module that helps you to avoid any unintentional violation of this Software License Agreement. Altova may use your internal network for license metering between installed versions of the Software.

(b) **Software Activation**. **Altova's Software may use your internal network and Internet connection for the purpose of transmitting license-related data at the time of installation, registration or update to an Altova-operated license server and validating the authenticity of the license-related data in order to protect Altova against unlicensed or illegal use of the Software and to improve customer service. Activation is based on the exchange of license related data between your computer and the Altova license server. You agree that Altova may use these measures and you agree to follow any applicable requirements.**

(c) **LiveUpdate**. Altova provides a new LiveUpdate notification service to you, which is free of charge. Altova may use your internal network and Internet connection for the purpose of transmitting license-related data to an Altova-operated LiveUpdate server to validate your license at appropriate intervals and determine if there is any update available for you.

(d) **Use of Data.** The terms and conditions of the Privacy Policy are set out in full at http://www.altova.com/privacy and are incorporated by reference into this Software License Agreement. By your acceptance of the terms of this Software License Agreement or use of the Software, you authorize the collection, use and disclosure of information collected by Altova for the purposes provided for in this Software License Agreement and/or the Privacy Policy as revised from time to time. European users understand and consent to the processing of personal information in the United States for the purposes described herein. Altova has the right in its sole discretion to amend this provision of the Software License Agreement and/or Privacy Policy at any time. You are encouraged to review the terms of the Privacy Policy as posted on the Altova Web site from time to time.

8. TERM AND TERMINATION

This Software License Agreement may be terminated (a) by your giving Altova written notice of termination; or (b) by Altova, at its option, giving you written notice of termination if you commit a breach of this Software License Agreement and fail to cure such breach within ten (10) days after notice from Altova. In addition the Software License Agreement governing your use

 © 2006 Altova GmbH

of a previous version that you have upgraded or updated of the Software is terminated upon your acceptance of the terms and conditions of the Software License Agreement accompanying such upgrade or update. Upon any termination of the Software License Agreement, you must cease all use of the Software that it governs, destroy all copies then in your possession or control and take such other actions as Altova may reasonably request to ensure that no copies of the Software remain in your possession or control. The terms and conditions set forth in Sections 1(g), (h), (i), 2, 5(b), (c), 9, and 10 survive termination as applicable.

9. RESTRICTED RIGHTS NOTICE AND EXPORT RESTRICTIONS

The Software was developed entirely at private expense and is commercial computer software provided with **RESTRICTED RIGHTS**. Use, duplication or disclosure by the U.S. Government or a U.S. Government contractor or subcontractor is subject to the restrictions set forth in this Agreement and as provided in FAR 12.211 and 12.212 (48 C.F.R. §12.211 and 12.212) or DFARS 227. 7202 (48 C.F.R. §227-7202) as applicable. Consistent with the above as applicable, Commercial Computer Software and Commercial Computer Documentation licensed to U.S. government end users only as commercial items and only with those rights as are granted to all other end users under the terms and conditions set forth in this Software License Agreement. Manufacturer is Altova GmbH, Rudolfsplatz, 13a/9, A-1010 Vienna, Austria/EU. You may not use or otherwise export or re-export the Software or Documentation except as authorized by United States law and the laws of the jurisdiction in which the Software was obtained. In particular, but without limitation, the Software or Documentation may not be exported or re-exported (i) into (or to a national or resident of) any U.S. embargoed country or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Department of Commerce's Table of Denial Orders. By using the Software, you represent and warrant that you are not located in, under control of, or a national or resident of any such country or on any such list.

10. GENERAL PROVISIONS

If you are located in the European Union and are using the Software in the European Union and not in the United States, then this Software License Agreement will be governed by and construed in accordance with the laws of the Republic of Austria (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the Handelsgericht, Wien (Commercial Court, Vienna) and you further agree and expressly consent to the exercise of personal jurisdiction in the Handelsgericht, Wien (Commercial Court, Vienna) in connection with any such dispute or claim.

If you are located in the United States or are using the Software in the United States then this Software License Agreement will be governed by and construed in accordance with the laws of the Commonwealth of Massachusetts, USA (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the federal or state courts of Massachusetts and you further agree and expressly consent to the exercise of personal jurisdiction in the federal or state courts of Massachusetts in connection with any such dispute or claim.

If you are located outside of the European Union or the United States and are not using the Software in the United States, then this Software License Agreement will be governed by and construed in accordance with the laws of the Republic of Austria (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the Handelsgericht, Wien (Commercial Court, Vienna) and you further agree and expressly consent to the exercise of personal jurisdiction in the Handelsgericht Wien (Commercial Court, Vienna) in connection with any such dispute or claim. This Software License Agreement will not be governed by the conflict of law rules of any jurisdiction or the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded.

This Software License Agreement contains the entire agreement and understanding of the parties with respect to the subject matter hereof, and supersedes all prior written and oral understandings of the parties with respect to the subject matter hereof. Any notice or other communication given under this Software License Agreement shall be in writing and shall have been properly given by either of us to the other if sent by certified or registered mail, return receipt requested, or by overnight courier to the address shown on Altova's Web site for Altova and the address shown in Altova's records for you, or such other address as the parties may designate by notice given in the manner set forth above. This Software License Agreement will bind and inure to the benefit of the parties and our respective heirs, personal and legal representatives, affiliates, successors and permitted assigns. The failure of either of us at any time to require performance of any provision hereof shall in no manner affect such party's right at a later time to enforce the same or any other term of this Software License Agreement. This Software License Agreement may be amended only by a document in writing signed by both of us. In the event of a breach or threatened breach of this Software License Agreement by either party, the other shall have all applicable equitable as well as legal remedies. Each party is duly authorized and empowered to enter into and perform this Software License Agreement. If, for any reason, any provision of this Software License Agreement is held invalid or otherwise unenforceable, such invalidity or unenforceability shall not affect the remainder of this Software License Agreement, and this Software License Agreement shall continue in full force and effect to the fullest extent allowed by law. The parties knowingly and expressly consent to the foregoing terms and conditions.

Last updated: 2005-05-05

© 2006 Altova GmbH

Index

.

.nt files,
exporting to, 87

A

AllDifferent,
creating collections of, 49
Anonymous classes,
toggling display on and off, 90

B

Base URI,
setting for document, 100
Blank nodes,
toggling display on and off, 90

C

Cardinality,
entering min and max on restrictions, 103
Cardinality constraints,
of properties, 38, 58
Checking the ontology,
syntax and semantics, 32, 54
Classes,
creating and deleting, 32
defining hierarchy, 54
defining relationships for, 34
enumerating instances for, 56
intersectionOf, 60
restrictions of, 60
unionOf, 60
Consistency check,
of ontologies, 90, 92
Copy,
description of command, 89
Copyright information, 108
Creating ontology classes, 32, 54
Cut,
description of command, 89

D

Datatype properties,
defining, 38, 58
Datatypes,
declaring XML Schema namespace for, 30
Delete,
description of command, 89
Deleting,
classes and other ontology items, 32
Detail View,
displaying graph in viewport, 16
configuring appearance, 95
description of, 13
Distinct members,
in allDifferent collections, 49
Distribution,
of Altova's software products, 108, 109, 111
Documentation,
overview, 6
Domain,
of properties, 38, 58
Dublin Core,
creating metadata with, 74
namespaces in RDF document, 71
using in RDF document, 71
Dublin Core vocabulary, 52

E

Edit menu,
description of commands in, 89, 90
Encoding,
of SemanticWorks documents, 87
setting defaults, 95
End User License Agreement, 108, 112
Enumerations,

© 2006 Altova GmbH

Enumerations,
of classes, 56
Errors Window,
description of, 17
in interface, 11
Evaluation period,
of Altova's software products, 108, 109, 111
Exporting files,
to Triples and XML formats, 87

F

Features,
of SemanticWorks, 9
File extensions,
for saving and exporting, 87
File menu,
description of commands in, 87
Find,
description of command, 89
Fonts,
configuring for Text View and RDF/OWL View, 95
Functional properties,
defining, 38

G

GUI,
broad description of, 11

H

Help menu,
description of, 102

I

Images,
saving Detail View graph as, 87
Importing namespaces,
for RDF documents, 97
Imports,
reloading, 92
Instances,
creating, 44, 56
defining predicates for, 44, 56
Interface,
broad description of, 11
IntersectionOf,
classes, 60
Introduction, 8

L

Language level,
selecting, 27
Legal information, 108
Level of ontology language,
selecting, 27
License, 112
information about, 108
License metering,
in Altova products, 110

M

Main Window,
in interface, 11, 13
Menu bar,
in interface, 11

N

Namespace imports,
for RDF documents, 97
Namespaces,
how to declare, 30
importing from ontology into RDF document, 66, 71

O

Object properties,

© 2006 Altova GmbH

Object properties,
defining, 38, 58
Objects,
in RDF statements, 68, 74
Onscreen help, 102
Ontologies,
for RDF document creation, 66
general creation and editing mechanism, 19
Ontology,
creating new, 27
language level, 27
Ontology level,
OWL DL tutorial, 52
Opening files in SemanticWorks, 87
Options dialog,
description, 95
Overview,
of categories in Main Window, 13
Overview Window,
description of, 16
in interface, 11
OWL DL,
checking syntax and semantics, 92
OWL DL ontology,
tutorial, 51
OWL Lite,
checking syntax and semantics, 92
OWL Lite ontology,
tutorial, 26

P

Paste,
description of command, 89
Predicates,
declaring in RDF statements, 68, 74
for instances, 44, 56
Prefixes,
declaring for namespaces, 30
Printing,
of documents, 87
Product features,
of SemanticWorks, 9
Properties,
cardinality constraints, 38, 58
declaring in RDF documents, 68, 74
defining, 38, 58
domain and range of, 38, 58
Property restrictions,
entering cardinality requirements, 103

R

Range,
of properties, 38, 58
RDF document,
and Dublin Core vocabulary, 71
RDF document creation,
from OWL DL ontology, 66
tutorial, 65
RDF documents,
creating new, 66
general creation and editing mechanism, 19
importing namespaces from ontologies, 97
referencing an ontology, 66, 71
RDF resources,
creating, 68, 74
RDF statements,
making, 68, 74
RDF/OWL level,
selecting, 92
RDF/OWL menu,
description of commands in, 92
RDF/OWL View,
description of, 13
Redo,
description of command, 89
Replace,
description of command, 89
Resources,
for RDF document from ontology, 66, 71
Restrictions,
of classes, 60
on properties, 103

S

Saving files in SemanticWorks, 87
Semantics check,
and Errors Window, 17

© 2006 Altova GmbH

Semantics check,
of ontology, 32, 54
SemanticWorks,
product features, 9
product information, 102
user manual, 3
Software product license, 112
Status Bar,
display of, 90
Support resources, 102
Syntax check,
and Errors Window, 17
of ontology, 32, 54

T

Text View,
description of, 13, 30
Toolbars,
in interface, 11
setting display of, 90
setting options for, 90
Tools menu,
description of commands in, 94
Triples files,
exporting to, 87
Tutorial,
OWL DL ontology, 51
OWL Lite ontology, 26
RDF document creation, 65

U

Undo,
description of command, 89
UnionOf,
classes, 60
URIref,
declaring prefixes of, 30
URIref prefixes,
declaring namespaces for, 99
expanding, 99
Usage,
overview, 19
Usage issues,
listing of, 103
User manual,
overview, 6

V

Valid,
result of semantics check, 92

W

Well-formed,
result of syntax check, 92
Window menu,
description of commands in, 101
Windows,
moving in interface, 11
Windows in GUI,
configuring display of, 101

X

XML files,
exporting to, 87
XML Schema,
declaring namespace for datatypes, 30
xml:base,
use of, 30

© 2006 Altova GmbH